

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-297570

(43)Date of publication of application : 18.11.1997

(51)Int. Cl. G09G 5/36
G09G 5/36
G06F 9/06
G06T 11/00
G09G 5/20

(21)Application number : 08-341801 (71)Applicant : CIRRUS LOGIC INC

(22)Date of filing : 20.12.1996 (72)Inventor : DEVIC GORAN

(30)Priority

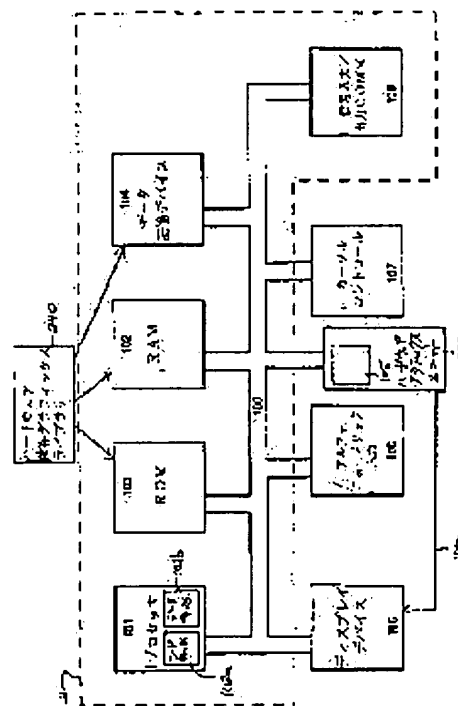
Priority number : 95 576872 Priority date : 21.12.1995 Priority country : US

(54) COMPUTER-CONTROLLED GRAPHICS DISPLAY SYSTEM AND METHOD OF PREPARING DISPLAY LIST IN IT

(57)Abstract:

PROBLEM TO BE SOLVED: To eliminate redesign of graphics library at a high level against a different kind of hardware.

SOLUTION: The method is performed by dividing the graphics library into a high-level graphics library which does not depend on hardware and a low-level graphics library which depends on hardware. When a high-level application issues the request list of the plotted graphics of a series of graphics primitives, based on this, an independent graphics rendering procedure constructs a batch array independent of hardware to store into a data cache 102b. A parametrizing procedure of hardware-dependent low-level graphics library 240 reads this batch cell in the code cache 102 and generates a microinstruction for a hardware unit 250 to add to the display list.



LEGAL STATUS

[Date of request for examination] 10.01.1997

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application]

other than the examiner's decision of
rejection or application converted
registration]

[Date of final disposal for application]

[Patent number] 3260090

[Date of registration] 14. 12. 2001

[Number of appeal against examiner's
decision of rejection]

[Date of requesting appeal against
examiner's decision of rejection]

[Date of extinction of right]

Copyright (C) ; 1998, 2003 Japan Patent Office

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平9-297570

(43)公開日 平成9年(1997)11月18日

(51)Int.Cl. ⁸	識別記号	庁内整理番号	F I	技術表示箇所
G 0 9 G 5/36	5 3 0		G 0 9 G 5/36	5 3 0 C
	5 1 0			5 1 0 V
G 0 6 F 9/08	4 1 0		G 0 6 F 9/08	4 1 0 H
G 0 6 T 11/00			G 0 9 G 5/20	
G 0 9 G 5/20			G 0 6 F 15/72	A

審査請求 有 請求項の数20 O L (全 32 頁)

(21)出願番号 特願平8-341801

(22)出願日 平成8年(1996)12月20日

(31)優先権主張番号 08/576, 872

(32)優先日 1995年12月21日

(33)優先権主張国 米国 (US)

(71)出願人 595158337

シーラス ロジック, インコーポレイテッド

Cirrus Logic, Inc.

アメリカ合衆国 カリフォルニア 94538,

フレモント, ウェスト ワレン アベニュー

- 3100

(72)発明者 グラン デビック

アメリカ合衆国 テキサス 78753, オ

ースティン, ドック ホリディ トレイ

ル 10401

(74)代理人 弁理士 山本 秀策

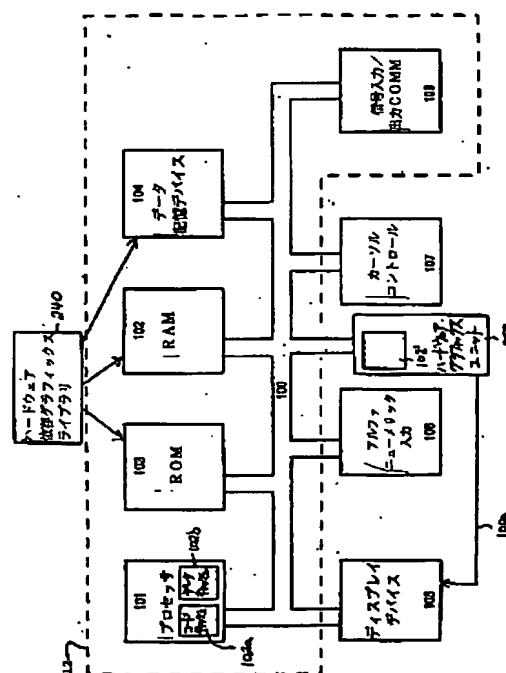
最終頁に続く

(54)【発明の名称】 コンピュータ制御されたグラフィックスディスプレイシステムおよび同システムにおいてディスプレイリストを作成する方法

(57)【要約】 (修正有)

【課題】 異種のハードウェアに対するグラフィックスライブラリのハイレベルでの再設計を不要とする。

【解決手段】 ハードウェア非依存のハイレベルグラフィックスライブラリとハードウェア依存ローレベルグラフィックスライブラリ240とに分ける。ハイレベルアプリケーションが一連のグラフィックスプリミティブの描画のリクエストを出すと、これに基づきハイレベルグラフィックスライブラリのハードウェア非依存グラフィックスレンダリングプロシージャがハードウェア非依存のパッチアレイを構築しデータキャッシュ102bに格納する。ハードウェア依存ローレベルグラフィックスライブラリ240のパラメータ化プロシージャはコードキャッシュ内102aにあつてこのパッチセルを読み出し、ハードウェアユニット250のためのマイクロインストラクションを生成し、ディスプレイリストに加える。



1

【特許請求の範囲】

【請求項 1】 バスに結合されたプロセッサと、
情報を格納するメモリユニットと、
該メモリユニットに格納されたディスプレイリストから
ハードウェア依存マイクロインストラクションを受け取り、
ディスプレイスクリーン上にイメージを生成するハード
ウェアグラフィックスユニットと、

該プロセッサによって実行されるハードウェア非依存
グラフィックスレンダリングプロシージャを含むハイレ
ベルグラフィックスライブラリであって、該ハードウェア
非依存グラフィックスレンダリングプロシージャが、ハ
イレベルアプリケーションからのグラフィックスレンダ
リングリクエストを処理することによって、グラフィッ
クスオペランドを含むハードウェア非依存出力データ構
造をつくる、ハイレベルグラフィックスライブラリと、
該プロセッサにより実行されるローレベルハードウェア
依存グラフィックスライブラリであって、該ハードウェア
非依存出力データ構造を処理することによって、該ハ
ードウェアグラフィックスユニット用の該マイクロイン
ストラクションを該データ構造から生成する、ローレベ
ルハードウェア依存グラフィックスライブラリと、を備
えている、コンピュータ制御されたグラフィックスディ
スプレイシステムであって、

該ハイレベルグラフィックスライブラリが、再設計しな
くともさまざまな種類の異なるハードウェアグラフィッ
クスユニットとコンパチブルである、システム。

【請求項 2】 前記ハードウェア非依存出力データ構造
が、複数のパッチセルのアレイを含んでおり、該複数の
パッチセルのそれぞれが、実行されるべき個別のグラフ
ィックス演算を表現しており、該パッチセルアレイが、
前記ローレベルハードウェア依存グラフィックスライ
ブラリにわたされシーケンシャルに処理されることによ
って、前記マイクロインストラクションを生成する、請
求項 1 に記載のシステム。

【請求項 3】 前記メモリユニットがコードキャッシュ
を含んでおり、前記ローレベルハードウェア依存グラフ
ィックスライブラリが、前記マイクロインストラクショ
ンを生成するために前記プロセッサによって実行される
パラメータ化プロシージャを含んでおり、前記パッチ
セルアレイが、キャッシュミスヒットを防止するために該
パラメータ化プロシージャにより割込みなしに処理され
る、請求項 2 に記載のシステム。

【請求項 4】 前記ローレベルハードウェア依存グラフ
ィックスライブラリが、ポリゴンプリミティブと、数セ
ットのグラフィックスラインと、数セットのグラフィッ
クスポイントとを処理するためのパラメータ化プロシー
ジャを含んでいる、請求項 2 に記載のシステム。

【請求項 5】 前記パラメータ化プロシージャが、さら
に、ビットレベル転送と、塗りつぶしと、テクスチャマ
ップフォーマット間の翻訳とを処理するためのものであ

2

る、請求項 4 記載のシステム。

【請求項 6】 前記ローレベルハードウェア依存グラフ
ィックスライブラリが、レンダリングパフォーマンスレ
ートを調整し、かつそれに対応して前記ディスプレイス
クリーン上に表示される前記イメージのレンダリングク
オリティを調整するために前記プロセッサによって実行
されるパフォーマンス／クオリティ調整プロシージャを
さらに含んでいる、請求項 2 に記載のシステム。

【請求項 7】 前記プロセッサによって実行される前記
パフォーマンス／クオリティ調整プロシージャが、グラフ
ィックスレンダリング用におこなわれる線形およびパ
ースペクティブ再分割レベルの調整と、ポリゴンオーバ
ーラップに用いられる誤差補正係数レベルの調整と、パ
ースペクティブレンダリングカットオフ用の閾値プリミ
ティブサイズの調整と、のためのものである、請求項 6
に記載のシステム。

【請求項 8】 アドレス／データベースに結合されたプロ
セッサと、
グラフィックス情報を格納するメモリユニットと、
ディスプレイリストに格納されたハードウェア依存マイ
クロインストラクションを処理し、かつ、該マイクロ
インストラクションに応答して、ディスプレイスクリー
ン上にイメージを生成するハードウェアグラフィックス
ユニットと、

該プロセッサによって実行されるハードウェア非依存
グラフィックスレンダリングプロシージャを有するハイレ
ベルグラフィックスライブラリであって、該ハードウェア
非依存グラフィックスレンダリングプロシージャが、
ハイレベルアプリケーションからのグラフィックスレン
ダリングリクエストを受け取り、該リクエストから複数
のパッチセルを含むハードウェア非依存アレイを生成す
るハイレベルグラフィックスライブラリであって、該複
数のパッチセルがそれぞれ、グラフィックスプリミティ
ブを含む個々のグラフィックスレンダリング演算を表現
している、ハイレベルグラフィックスライブラリと、
該グラフィックスレンダリングプロシージャから該複
数のパッチセルを含む該アレイを受け取るために、該プロ
セッサにより実行されるローレベルハードウェア依存
グラフィックスライブラリであって、該複数のパッチ
セルを含む該アレイの該複数のセルをパラメータ化す
ることによって、該マイクロインストラクションを該
ハードウェアグラフィックスユニットに対して生成する、
ローレベルハードウェア依存グラフィックスライブラリ
と、を備えている、コンピュータ制御されたグラフィッ
クスディスプレイシステム。

【請求項 9】 前記メモリユニットがコードキャッシュ
と、データキャッシュとを備えており、前記ローレベル
ハードウェア依存グラフィックスライブラリが、該コー
ドキャッシュに格納されているパラメータ化プロシー
ジャであって、該データキャッシュに格納されている前記

3

複数のパッチセルを含む前記アレの該複数のセルに対して、前記マイクロインストラクションを生成するように実行されるパラメータ化プロシーダを含んでおり、該複数のパッチセルを含む該アレの該複数のセルが、コードキャッシュミスヒットを防止するために該パラメータ化プロシーダの割込みなしに処理される、請求項 8 に記載のシステム。

【請求項 10】 前記ローレベルハードウェア依存グラフィックスライブラリが、ポリゴンプリミティブと、数セットのグラフィックスラインと、数セットのグラフィックスポイントとを処理するために前記プロセッサによって実行されるパラメータ化プロシーダを含んでおり、請求項 8 に記載のシステム。

【請求項 11】 前記パラメータ化プロシーダが、さらに、ビットレベル転送と、塗りつぶしとを処理し、テクスチャマップフォーマット間の翻訳をおこなうためのものである、請求項 10 に記載のシステム。

【請求項 12】 前記ローレベルハードウェア依存グラフィックスライブラリが、レンダリングパフォーマンスレートを調整し、かつそれに対応して前記ディスプレイスクリーン上に表示される前記グラフィックスイメージのレンダリングクオリティを調整するために前記プロセッサによって実行されるパフォーマンス／クオリティ調整プロシーダをさらに含んでいる、請求項 8 に記載のシステム。

【請求項 13】 前記プロセッサによって実行される前記パフォーマンス／クオリティ調整プロシーダが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割レベルの調整と、ポリゴンオーバーラップに用いられる誤差補正係数レベルの調整と、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズの調整と、のためのものである、請求項 12 に記載のシステム。

【請求項 14】 バスに結合されたプロセッサと、情報を格納するメモリユニットと、ディスプレイリスト内のマイクロインストラクションに基づいて、ディスプレイスクリーン上にイメージをレンダリングするハードウェアグラフィックスユニットと、を含んでいる、コンピュータ制御されたグラフィックスシステムにおいて、該ディスプレイリストを作成する方法であって、該コンピュータによってインプリメントされるステップである、該プロセッサによって実行されるハイレベルアプリケーションを用いて、ポリゴン、ラインおよびポイントを含むグラフィックスプリミティブをレンダリングするリクエストを含む、1 セットのグラフィックスレンダリングリクエストを生成するステップと、該プロセッサによって実行されるハイレベルグラフィックスライブラリのハードウェア依存プロシーダを用いて、該 1 セットのグラフィックスレンダリングリクエストを、複数のパッチセルを含むハードウェア非依存アレ

4

イに翻訳するステップであって、該複数のパッチセルがそれぞれ、個々のグラフィックス演算を含んでいる、ステップと、

該複数のパッチセルを含む該ハードウェア非依存アレを受け取り、かつ、ローレベルハードウェア依存グラフィックスライブラリを用いて該複数のパッチセルを含む該アレの該複数のセルをシーケンシャルに処理することによって、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレから生成するステップであって、該ローレベルハードウェア依存グラフィックスライブラリが該プロセッサによって実行される、ステップと、

該ディスプレイリストにアクセスし、該ハードウェアグラフィックスユニットを用いて該ディスプレイスクリーン上にイメージを表示するステップと、を含んでいる方法であって、

該ハイレベルグラフィックスライブラリが、再設計をしなくても、多種多様の異なるハードウェアグラフィックスユニットとコンパチブルである、方法。

【請求項 15】 前記複数のパッチセルを含む前記ハードウェア非依存アレを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレから生成する前記ステップが、該複数のパッチセル内でポリゴンプリミティブを処理することによって、該プリミティブからディスプレイリストマイクロインストラクションを生成するステップと、該複数のパッチセル内で数セットのラインを処理することによって、該ラインからディスプレイリストマイクロインストラクションを生成するステップと、該複数のパッチセル内で数セットのポイントを処理することによって、該ポイントからディスプレイリストマイクロインストラクションを生成するステップと、をさらに含んでいる、請求項 14 に記載の方法。

【請求項 16】 前記複数のパッチセルを含む前記ハードウェア非依存アレを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレから生成する前記ステップが、ビットレベル転送を処理することによって、該ビットレベル転送からディスプレイリストマイクロインストラクションを生成するステップと、塗りつぶし演算を処理することによって、該演算からディスプレイリストマイクロインストラクションを生成するステップと、

テクスチャマップ翻訳を処理することによって、テクスチャマップをあるディスプレイフォーマットから別のディスプレイフォーマットへと翻訳するステップと、をさらに含んでいる、請求項 15 に記載の方法。

【請求項 17】 前記複数のパッチセルを含む前記ハードウェア非依存アレを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディス

5

レイリストを該アレイから生成する前記ステップが、前記ハードウェア依存グラフィックスライブラリのパラメータ化プロシーダをキャッシュメモリユニット内へとロードするステップと、該複数のパッチセルを含む該アレイを処理しながら、該キャッシュユニットからの該パラメータ化プロシーダを、割込みなしに該複数のパッチセルを含む該アレイの該複数のセルに対してシーケンシャルに実行することによって、キャッシュミスヒットを防止するステップと、を含んでいる、請求項 11 に記載の方法。

【請求項 18】 前記ディスプレイスクリーン上にレンダリングされたパフォーマンス/クオリティコントロールパネルの設定を調整するステップと、該設定に基づき、前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスレートを増加あるいは減少させるステップと、それに対応して、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させるステップと、をさらに含んでいる、請求項 15 に記載の方法。

【請求項 19】 前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスを増加あるいは減少させる前記ステップ、および、それに対応して、前記設定に基づき、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させる前記ステップが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割のレベルを調整するステップと、ポリゴンオーバーラップ時に用いられる誤差補正係数のレベルを調整するステップと、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズを調整するステップと、をさらに含んでいる、請求項 18 に記載の方法。

【請求項 20】 前記グラフィックスレンダリングリクエストが、テクスチャマップのレンダリング演算をさらに含んでいる、請求項 15 に記載の方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、コンピュータ制御されたグラフィックスディスプレイシステムの分野に関する。具体的には、本発明は、ハードウェアグラフィックスアクセラレータをインタフェースするソフトウェアプログラムに関する。

【0002】 に関する。

【0003】

【従来の技術】 高い品質および高いパフォーマンスのコンピュータ制御されたグラフィックスレンダリングシステムは、グラフィックス情報を高速で処理するための専用の電子回路ボードに大きく依存する。これらの専用電

6

子回路ボードは、「グラフィックスアクセラレータ」または「グラフィックスハードウェアユニット」ともよばれている。グラフィックスアクセラレータは、この技術分野では知られているように、グラフィックスプリミティブ（つまりライン、ポリゴン、トライアングル、シェーディングされた（shaded）またはアルファブレンディングされた（alpha blended）トライアングルなど）に伴うグラフィックスデータを処理して、コンピュータディスプレイユニット上にイメージをレンダリングすることができるように特別に設計されている。グラフィックスアクセラレータに与えられるグラフィックスデータは、グラフィックスアクセラレータに認識できるハードウェア依存フォーマット（「ハードウェア依存グラフィックスデータ」）のかたちで与えられる。このハードウェア依存グラフィックスデータは、典型的には、コンピュータメモリの中にディスプレイリストのかたちで生成される。

【0004】 マルチレイヤグラフィックスシステム内では、グラフィックスアクセラレータは、画素を操作してコンピュータディスプレイ上にイメージをレンダリングするために最も低いレベルのレイヤで動作する。グラフィックスアクセラレータは、ディスプレイリスト内のマイクロインストラクションをコンピュータディスプレイ上のイメージに翻訳する低いレベルのグラフィックス演算をおこなう。より高いレベルのレイヤでは、グラフィックスシステム内で実行されるソフトウェアプログラム（「アプリケーション」）は、ある種のイメージを表示するリクエストを生成する。これらのリクエストは典型的には、中間のハイレベルのグラフィックスライブラリにわたされ、それからレンダリングのためにグラフィックスアクセラレータにわたされる、（例えばイメージを表現する）ハードウェア非依存のグラフィックスプリミティブのリストを含んでいる。ハイレベルグラフィックスライブラリは、多数のグラフィックスコマンドやグラフィックスの形をサポートしており、典型的には、アプリケーションのリクエストを直接にハードウェア依存のディスプレイリストに変換するプロシーダを含んでいる。

【0005】 グラフィックスアクセラレータは、ハードウェアデバイスなので、設計そのものによって左右され、ベンダによっても左右される。換言すれば、異なるグラフィックスアクセラレータは、異なるグラフィックスデータフォーマット（例えば異なるデータ構造、異なるグラフィックスレンダリングコード、異なるメモリパーティションおよびロケーションなど）を用い、かつ異なるキャッシュ割り当てを用いて動作する。

【0006】 図 1 は、上述のような従来技術によるグラフィックスディスプレイシステム 5 を示している。システム 5 は、デジタルコンピュータシステム内で実行可能なハイレベルアプリケーションプログラム 10 を含

7

む。アプリケーションプログラム10は、アプリケーション10からのグラフィックスディスプレイリクエストをハードウェア依存ディスプレイリストに翻訳するためのハイレベルグラフィックスライブラリ12および14とのインタフェースをおこなう。2つのよく知られているグラフィックスライブラリは、3次元ダイナミックデバイスドライバインタフェース（「3D DDI」）ライブラリ12およびオープングラフィックスライブラリ（「OpenGL」）ライブラリ14である。これらのライブラリ12および14は、インタフェース16を介してグラフィックスハードウェアユニット18（例えばアクセラレータボード）と直接にインタフェースする、ハードウェア特定のレンダリングルーチン（「プロシージャ」）をいくつか含んでいる。インタフェース16は、ハードウェアユニット18をトランスペアレントにはせず、ライブラリ12および14と、ハードウェアユニット18との間のコミュニケーションプロトコルを促進するはたらきをするだけである。ライブラリ12および14のプロシージャは、アプリケーションプログラム10からのグラフィックスレンダリングリクエストを受け取る。グラフィックスハードウェアユニット18は、コンピュータスクリーン20に結合されており、その上にイメージをレンダリングする。別々のものとして示されているが、コンパイルおよびリンキングのあいだは、図1のライブラリ12および14の要求されたプロシージャは、典型的にはハイレベルアプリケーション10のインストラクションを構成するように含まれている。アプリケーション10は、グラフィックスライブラリ12および14以外の多くの異なるライブラリから生じるインストラクションをも含む。

【0007】

【発明が解決しようとする課題】グラフィックスライブラリ12および14のプロシージャは、非常に構造的で、ハードウェアに依存した通信プロトコルおよびデータ構造を生成することによって、ハイレベルアプリケーション10がグラフィックスハードウェアユニット18と（ハードウェア非依存データ構造を用いて）通信することができるようにする。グラフィックスライブラリ12および14は、ハードウェア依存ではあるが、アプリケーション10に対してハードウェアトランスペアレントなインタフェースを提供する、非常に高いレベルのハードウェア依存ソフトウェアルーチンのセットを含む。ライブラリ12および14は高いレベルなので、さまざまな種類の複雑なグラフィックス形状およびディスプレイオプションを広くサポートできる。ある特定のハードウェアユニット18をサポートできるようにライブラリ12または14のどちらかをインプリメントするためには、ライブラリ関数および形状のすべては、選ばれたハードウェアユニット18の特定のフォーマットでインプリメントされなければならない。よって、これらのハイ

8

レベルグラフィックスライブラリ12および14は、それらがサポートする異なるハードウェアユニット18それぞれに対して、大幅に再設計および再コーディングされなければならない。これは、グラフィックスシステムのデザイナーにとっては望ましくないことである。なぜなら、異なるハードウェアユニット18をサポートするためには、ソフトウェアの再設計および再コーディングを大規模におこなうことが、それぞれのライブラリ12および14に要求されるからである。グラフィックスハードウェアユニット18における各種変更および変動に対してもっと簡単に適応可能なグラフィックスシステムを提供することが望ましい。特に、異なるいくつかのグラフィックスハードウェアユニット18に適用するときに、ハイレベルグラフィックスライブラリ12および14の再設計および再コーディングを必要としないソフトウェアシステムを提供することが望ましい。

【0008】図1および図2を参照する。図2は、グラフィックスプリミティブを処理するための、従来技術のコンピュータによりインプリメントされる処理30のフローチャートである。この処理30は、従来技術によるコンピュータ制御されたグラフィックスディスプレイシステム5（図1）内でインプリメントされる。処理30は、ブロック32からスタートする。ブロック32において、ハイレベルアプリケーションプログラム10は、個々のグラフィックスプリミティブ（例えば、ライン、ポリゴン、トライアングルなど）を表現するデータ構造をディスプレイ20上にレンダリングすることをリクエストする。ブロック34では、アプリケーション10は、グラフィックスプリミティブを表現するハードウェア非依存データ構造を、グラフィックスライブラリ12または14のプロシージャにわたす。ブロック36では、グラフィックスライブラリ12または14は、グラフィックスプリミティブのデータ構造を、ハードウェアユニット18特定の、1セットのローレベルマイクロインストラクションに翻訳する。これらのマイクロインストラクションは、ハードウェアユニット18が判読可能な「ディスプレイリスト」の一部であることが多い。ブロック38では、ハードウェアユニット18はディスプレイリストにアクセスし、グラフィックスプリミティブをディスプレイスクリーン上20にレンダリングする。アプリケーションプログラム10から生ずる後続プリミティブも、引き続き同様に処理される。なぜなら、アプリケーションプログラム10は、典型的には、完全なイメージをレンダリングするためには、複数のグラフィックスプリミティブを要求するからである。

【0009】図2の処理30は、グラフィックスシステム5内のメモリリソースを効率的に用いていない。なぜならブロック32からのそれぞれのプリミティブは、ハイレベルアプリケーション10とハードウェアユニット18との間で（例えば、ブロック32と38との間で）

9

シリアルに処理されていくからである。特に単一のグラフィックスプリミティブのレンダリング処理をおこなうときには、アプリケーション 10 内のプロシージャがまず実行された後、グラフィックスライブラリ (12 または 14) のプロシージャが実行される。このような処理は、後続のプリミティブについても繰り返される。この状態では、コードおよびデータキャッシュ内に不連続性 (disjunction) が発生する。なぜなら異なる情報およびインストラクションが、(1) ハイレベルアプリケーション 10 のプロシージャの実行中、そして (2) グラフィックスライブラリ 12 または 14 のプロシージャの実行中にこれらのキャッシュユニットを通るからである。その結果、1 セットのグラフィックスプリミティブをレンダリングするあいだに、データのキャッシュミスヒットおよびコードのキャッシュミスヒットが何回も起こる。よって、アプリケーションプログラム 10 から発する 1 セットのグラフィックスプリミティブから構成されるイメージをレンダリングするときに、より効率的に動作するグラフィックスレンダリング処理を提供できれば効果的である。

【0010】従来技術によれば、上述の課題があった。本発明は、上記課題を解決するためになされたものであり、その目的とするところは、ハイレベルグラフィックスライブラリの再設計および再コーディングを必要とせず、異種のハードウェアユニットに対して容易に適用可能であるグラフィックスシステムを提供することにある。さらに本発明の他の目的は、アプリケーションプログラムから発する複数のグラフィックスプリミティブからイメージをレンダリングするために、メモリリソースを効率的に利用できるグラフィックスシステムを提供することである。本発明によるこれらの効果およびその他の効果は、以下に述べる本発明の説明により明らかになるだろう。

【0011】

【課題を解決するための手段】本発明によるコンピュータ制御されたグラフィックスディスプレイシステムは、バスに結合されたプロセッサと、情報を格納するメモリユニットと、該メモリユニットに格納されたディスプレイリストからハードウェア依存マイクロインストラクションを受け取り、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、該プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを含むハイレベルグラフィックスライブラリであって、該ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングクエストを処理することによって、グラフィックスオペランドを含むハードウェア非依存出力データ構造をつくる、ハイレベルグラフィックスライブラリと、該プロセッサにより実行されるローレベルハードウェア依

10

存グラフィックスライブラリであって、該ハードウェア非依存出力データ構造を処理することによって、該ハードウェアグラフィックスユニット用の該マイクロインストラクションを該データ構造から生成する、ローレベルハードウェア依存グラフィックスライブラリと、を備えている、コンピュータ制御されたグラフィックスディスプレイシステムであって、該ハイレベルグラフィックスライブラリが、再設計しなくてもさまざまな種類の異なるハードウェアグラフィックスユニットとコンパチブルであることにより上記目的が達成される。

【0012】ある実施形態では、前記ハードウェア非依存出力データ構造が、複数のバッチセルのアレイを含んでおり、該複数のバッチセルのそれぞれが、実行されるべき個別のグラフィックス演算を表現しており、該バッチセルアレイが、前記ローレベルハードウェア依存グラフィックスライブラリにわたされシーケンシャルに処理されることによって、前記マイクロインストラクションを生成する。

【0013】ある実施形態では、前記メモリユニットがコードキャッシュを含んでおり、前記ローレベルハードウェア依存グラフィックスライブラリが、前記マイクロインストラクションを生成するために前記プロセッサによって実行されるパラメータ化プロシージャを含んでおり、前記バッチセルアレイが、キャッシュミスヒットを防止するために該パラメータ化プロシージャにより割込みなしに処理される。

【0014】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、ポリゴンプリミティブと、数セットのグラフィックスラインと、数セットのグラフィックスポイントとを処理するためのパラメータ化プロシージャを含んでいる。

【0015】ある実施形態では、前記パラメータ化プロシージャが、さらに、ビットレベル転送と、塗りつぶしと、テクスチャマップフォーマット間の翻訳とを処理するためのものである。

【0016】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、レンダリングパフォーマンスレートを調整し、かつそれに対応して前記ディスプレイスクリーン上に表示される前記イメージのレンダリングクオリティを調整するために前記プロセッサによって実行されるパフォーマンス／クオリティ調整プロシージャをさらに含んでいる。

【0017】ある実施形態では、前記プロセッサによって実行される前記パフォーマンス／クオリティ調整プロシージャが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割レベルの調整と、ポリゴンオーバーラップに用いられる誤差補正係数レベルの調整と、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズの調整と、のためのものである。

11

【0018】本発明によるコンピュータ制御されたグラフィックスディスプレイシステムは、アドレス／データバスに結合されたプロセッサと、グラフィックス情報を格納するメモリユニットと、ディスプレイリストに格納されたハードウェア依存マイクロインストラクションを処理し、かつ、該マイクロインストラクションにตอบสนองして、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、該プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを有するハイレベルグラフィックスライブラリであって、該ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングリクエストを受け取り、該リクエストから複数のパッチセルを含むハードウェア非依存アレイを生成するハイレベルグラフィックスライブラリであって、該複数のパッチセルがそれぞれ、グラフィックスプリミティブを含む個々のグラフィックスレンダリング演算を表現している、ハイレベルグラフィックスライブラリと、該グラフィックスレンダリングプロシージャから該複数のパッチセルを含む該アレイを受け取るために、該プロセッサにより実行されるローレベルハードウェア依存グラフィックスライブラリであって、該複数のパッチセルを含む該アレイの該複数のセルをパラメータ化することによって、該マイクロインストラクションを該ハードウェアグラフィックスユニットに対して生成する、ローレベルハードウェア依存グラフィックスライブラリと、を備えており、そのことにより上記目的が達成される。

【0019】ある実施形態では、前記メモリユニットがコードキャッシュと、データキャッシュとを備えており、前記ローレベルハードウェア依存グラフィックスライブラリが、該コードキャッシュに格納されているパラメータ化プロシージャであって、該データキャッシュに格納されている前記複数のパッチセルを含む前記アレイの該複数のセルに対して、前記マイクロインストラクションを生成するように実行されるパラメータ化プロシージャを含んでおり、該複数のパッチセルを含む該アレイの該複数のセルが、コードキャッシュミスヒットを防止するために該パラメータ化プロシージャの割込みなしに処理される。

【0020】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、ポリゴンプリミティブと、数セットのグラフィックスラインと、数セットのグラフィックスポイントとを処理するために前記プロセッサによって実行されるパラメータ化プロシージャを含んでいる。

【0021】ある実施形態では、前記パラメータ化プロシージャが、さらに、ビットレベル転送と、塗りつぶしとを処理し、テクスチャマップフォーマット間の翻訳をおこなうためのものである。

12

【0022】ある実施形態では、前記ローレベルハードウェア依存グラフィックスライブラリが、レンダリングパフォーマンスレートを調整し、かつそれに対応して前記ディスプレイスクリーン上に表示される前記グラフィックスイメージのレンダリングクオリティを調整するために前記プロセッサによって実行されるパフォーマンス／クオリティ調整プロシージャをさらに含んでいる。

【0023】ある実施形態では、前記プロセッサによって実行される前記パフォーマンス／クオリティ調整プロシージャが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割レベルの調整と、ポリゴンオーバーラップに用いられる誤差補正係数レベルの調整と、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズの調整と、のためのものである。

【0024】本発明によるコンピュータ制御されたグラフィックスシステムにおいて、ディスプレイリストを作成する方法は、バスに結合されたプロセッサと、情報を格納するメモリユニットと、ディスプレイリスト内のマイクロインストラクションに基づいて、ディスプレイスクリーン上にイメージをレンダリングするハードウェアグラフィックスユニットと、を含んでいる、コンピュータ制御されたグラフィックスシステムにおいて、該ディスプレイリストを作成する方法であって、該コンピュータによってインプリメントされるステップである、該プロセッサによって実行されるハイレベルアプリケーションを用いて、ポリゴン、ラインおよびポイントを含むグラフィックスプリミティブをレンダリングするリクエストを含む、1セットのグラフィックスレンダリングリクエストを生成するステップと、該プロセッサによって実行されるハイレベルグラフィックスライブラリのハードウェア依存プロシージャを用いて、該1セットのグラフィックスレンダリングリクエストを、複数のパッチセルを含むハードウェア非依存アレイに翻訳するステップであって、該複数のパッチセルがそれぞれ、個々のグラフィックス演算を含んでいる、ステップと、該複数のパッチセルを含む該ハードウェア非依存アレイを受け取り、かつ、ローレベルハードウェア依存グラフィックスライブラリを用いて該複数のパッチセルを含む該アレイの該複数のセルをシーケンシャルに処理することによって、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを、該アレイから生成するステップであって、該ローレベルハードウェア依存グラフィックスライブラリが該プロセッサによって実行される、ステップと、該ディスプレイリストにアクセスし、該ハードウェアグラフィックスユニットを用いて該ディスプレイスクリーン上にイメージを表示するステップと、を含んでいる方法であって、該ハイレベルグラフィックスライブラリが、再設計をしなくても、多種多様の異なるハードウェアグラフィックスユニットとコンパチブルで

13

あり、そのことにより上記目的が達成される。

【0025】ある実施形態では、前記複数のパッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、該複数のパッチセル内でポリゴンプリミティブを処理することによって、該プリミティブからディスプレイリストマイクロインストラクションを生成するステップと、該複数のパッチセル内で数セットのラインを処理することによって、該ラインからディスプレイリストマイクロインストラクションを生成するステップと、該複数のパッチセル内で数セットのポイントを処理することによって、該ポイントからディスプレイリストマイクロインストラクションを生成するステップと、をさらに含んでいる。

【0026】ある実施形態では、前記複数のパッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、ビットレベル転送を処理することによって、該ビットレベル転送からディスプレイリストマイクロインストラクションを生成するステップと、塗りつぶし演算を処理することによって、該演算からディスプレイリストマイクロインストラクションを生成するステップと、テクスチャマップ翻訳を処理することによって、テクスチャマップをあるディスプレイフォーマットから別のディスプレイフォーマットへと翻訳するステップと、をさらに含んでいる。

【0027】ある実施形態では、前記複数のパッチセルを含む前記ハードウェア非依存アレイを受け取り、かつ、複数のマイクロインストラクションを含むハードウェア依存ディスプレイリストを該アレイから生成する前記ステップが、前記ハードウェア依存グラフィックスライブラリのパラメータ化プロシージャをキャッシュメモリユニット内へとロードするステップと、該複数のパッチセルを含む該アレイを処理しながら、該キャッシュユニットからの該パラメータ化プロシージャを、割込みなしに該複数のパッチセルを含む該アレイの該複数のセルに対してシーケンシャルに実行することによって、キャッシュミスヒットを防止するステップと、を含んでいる。

【0028】ある実施形態では、前記ディスプレイスクリーン上にレンダリングされたパフォーマンス/クオリティコントロールパネルの設定を調整するステップと、該設定に基づき、前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスレートを増加あるいは減少させるステップと、それに対応して、該設定に基づき、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させるステップと、をさらに含んでいる。

14

【0029】ある実施形態では、前記ハードウェアグラフィックスユニットのレンダリングパフォーマンスを増加あるいは減少させる前記ステップ、および、それに対応して、前記設定に基づき、該ハードウェアグラフィックスユニットのレンダリングクオリティを向上あるいは低下させる前記ステップが、グラフィックスレンダリング用におこなわれる線形およびパースペクティブ再分割のレベルを調整するステップと、ポリゴンオーバーラップ時に用いられる誤差補正係数のレベルを調整するステップと、パースペクティブレンダリングカットオフ用の閾値プリミティブサイズを調整するステップと、をさらに含んでいる。

【0030】ある実施形態では、前記グラフィックスレンダリングリクエストが、テクスチャマップのレンダリング演算をさらに含んでいる。

【0031】以下に作用を説明する。

【0032】本発明によれば、ローレベルハードウェア依存グラフィックスライブラリが、（好ましくはハードウェア非依存である）ハイレベルグラフィックスライブラリと、グラフィックスアクセラレータハードウェアユニットとの間のインタフェースを提供する。このローレベルハードウェア非依存ライブラリは、グラフィックスアクセラレータと直接に通信し、かつ、ハイレベルのハードウェア非依存グラフィックスライブラリをほとんど、または全く変更することを必要とせずに、比較的少量の再コーディングをおこなうだけで、異種のグラフィックスハードウェアユニットに適応可能である、比較的ローレベルのインタフェースを提供する。ローレベルハードウェア依存ライブラリプロシージャは、低速処理だが高画質なイメージレンダリングと、低画質だが高速なイメージレンダリングとのあいだで調整可能なクオリティメータ（quality meter）を提供する。ローレベルハードウェア依存ライブラリプロシージャは、複数のパッチセルから構成されるアレイを受け取ることによってパッチ処理をおこなう。それぞれのパッチセルは、別々のプリミティブを有する。アレイは、ある設定値でローレベルハードウェア依存ライブラリプロシージャにわたされた後、シーケンシャルに処理される。この構成によって、アレイのパラメータ化（parameterization）のときに（例えばパラメータ化ルーチンは、標準的なコードキャッシュ、例えば8kのものの中に納まる）インストラクションキャッシュのミスヒットを確実に防止でき、データキャッシュのミスヒットも確実に最小限にとどめることができる。また、ローレベルハードウェア依存ライブラリプロシージャは、異なるテクスチャマップングデータフォーマット間での自動翻訳を可能にする。その結果、RGB-アルファフォーマットが使用可能となるか、またはカラーパレットへのインデックスを用いるフォーマットが使用可能となる。ローレベルのインタフェースを提供することによって、本発明は、グラフィ

15

ックスライブラリに改変を施すことを必要とせずに、異なるハードウェアグラフィックスアクセラレータに対して容易に適応可能であるシステムを提供する。

【0033】具体的には、本発明の実施形態においては、バスに結合されたプロセッサと、プロセッサと連携して動作する、情報格納のためのメモリユニットと、メモリユニットに格納されたディスプレイリストからハードウェア依存マイクロインストラクションを受け取り、ディスプレイスクリーン上にイメージを生成するハードウェアグラフィックスユニットと、プロセッサによって実行されるハードウェア非依存グラフィックスレンダリングプロシージャを含むハイレベルグラフィックスライブラリであって、ハードウェア非依存グラフィックスレンダリングプロシージャが、ハイレベルアプリケーションからのグラフィックスレンダリングリクエストを処理することによってグラフィックスオペランドを含むハードウェア非依存出力データ構造をつくる、ハイレベルグラフィックスライブラリと、ハードウェア非依存出力データ構造を処理することによって、その構造からハードウェアグラフィックスユニット用のマイクロインストラクションを生成する、プロセッサによって実行されるローレベルハードウェア依存グラフィックスライブラリと、を有するコンピュータ制御されたグラフィックスディスプレイシステムが提供される。このグラフィックスディスプレイシステムにおいては、ハイレベルグラフィックスライブラリは、再設計しなくてもさまざまな種類の異なるハードウェアグラフィックスユニットと互換性であり、ハイレベルグラフィックスライブラリからのハードウェア非依存出力データ構造は、複数のパッチセルから構成されるアレイを有している。それぞれのパッチセルは、実行されるべき個別のグラフィックス演算を表現しており、パッチセルのアレイは、ローレベルハードウェア依存グラフィックスライブラリにわたされて、マイクロインストラクションを生成するためにシーケンシャルに処理される。

【0034】また、他の実施形態においては、上記特徴に加えて、ローレベルハードウェア依存グラフィックスライブラリは、ポリゴンプリミティブ、数セットのグラフィックスライン、および数セットのグラフィックスポイントを処理するためのパラメータ化プロシージャを備えており、パラメータ化プロシージャは、さらにビットレベル転送、塗りつぶし、およびテクスチャマッピングフォーマット間での翻訳を処理するためのものである。

【0035】また、他の実施形態においては、上記特徴に加えて、ローレベルハードウェア依存グラフィックスライブラリは、さらにレンダリングパフォーマンスレートを調整し、かつ、それに応じてディスプレイスクリーン上に表示されるイメージのレンダリングクオリティを調整するための、プロセッサによって実行されるパフォーマンス/クオリティ調整プロシージャを含んでいる。

16

また、他の実施形態では、上記構成に基づいて、ディスプレイリストを生成する方法が提供される。

【0036】本発明の構成要素には、ハードウェア非依存グラフィックスライブラリと、グラフィックスハードウェアユニットとの間のローレベルハードウェア依存グラフィックスライブラリ（結合ライブラリ）とがある。結合ライブラリプロシージャは、グラフィックスハードウェアユニットに直接結合する比較的ローレベルのインタフェースを提供し、比較的少量しか再コーディングしなくても、異なるグラフィックスハードウェアユニットに適応できる。このときハードウェア非依存グラフィックスライブラリは、全く変更する必要がない。結合ライブラリプロシージャは、低速処理だが高画質なイメージレンダリングと、低画質だが高速なイメージレンダリングとのあいだで調整可能なクオリティメータを提供する。結合ライブラリプロシージャは、複数のパッチセルから構成されるアレイを受け取ることによって、パッチ処理をおこなう。それぞれのパッチセルは、別々のプリミティブを有する。パッチアレイは、ある設定値で結合ライブラリプロシージャにわたされた後、シーケンシャルに処理される。この構成によって、アレイのパラメータ化のときの（例えば、パラメータ化ルーチンは、標準的なコードキャッシュ中に納まる）インストラクションキャッシュのミスヒットを確実に防止でき、データキャッシュのミスヒットも確実に最小限にとどめることができる。また結合ライブラリプロシージャは、異なるテクスチャマッピングデータフォーマット間での自動翻訳を可能にする。その結果、RGB-アルファフォーマットが使用可能となるか、またはカラーパレットへのインデックスを用いるフォーマットが使用可能となる。ローレベルのインタフェースを提供することによって、本発明は、グラフィックスライブラリに改変を施すことを必要とせずに、異なるハードウェアグラフィックスアクセラレータに対して容易に適応可能であるシステムを提供する。

【0037】

【発明の実施の形態】本発明の以下の詳細な説明においては、本発明を十分に理解できるようにするために、多くの具体的な詳細が述べられる。しかし本発明が、これらの具体的な詳細から離れて、その代替要素や方法を用いることによって実施できることは、当業者には明らかであろう。その他の例については、本発明の態様が不必要に不明瞭にならない限り、よく知られている方法、プロシージャ、要素、および回路などについてはその詳細を割愛する。

【0038】（表記法および用語のつかいかた）以下の詳細な説明の中には、プロシージャ、論理ブロック、処理、およびその他のコンピュータメモリ内のデータビットに対する演算のシンボリックな表現に関して述べられるものもある。これらの説明および表現は、データ処理

17

技術の分野の当業者によって用いられる手段であり、その作業の意味内容を最も効率的に他の当業者に伝えるためのものである。プロシージャ、論理ブロック、処理などは、ここでは一般に、所望の結果を得るための矛盾のないステップまたはインストラクションのシーケンスとしてとらえられている。これらのステップは、物理的な量を物理的に操作することを要求するものである。必ずそうであるわけではないが、通常、これらの物理的な操作は、格納し、転送し、統合し、比較できるか、または

10 そうでなければコンピュータシステムにおいて操作できる電氣的または磁氣的な信号のかたちをとる。簡単のために、また慣用法に基づいて、本発明についてはこれらの信号は、ビット、値、要素、シンボル、文字、語、数などのかたちで参照される。

【0039】しかしこれらの話すべては、物理的な操作および量を参照するものとして解釈されるべきであり、単に簡単のために付したラベルに過ぎず、この分野で通常用いられる用語に鑑みて解釈されるべきであることには注意されたい。特にそうではないことを述べない限り、以下の説明から明らかなように、本発明の説明を通じて、「処理」、「コンピューティング」、「計算」、「決定」、「表示」などの語を利用して議論するとき

20 は、データを操作し、変換するコンピュータシステム、または同様の電子的なコンピューティングデバイスの動作および処理をさすものとする。データは、コンピュータシステムのレジスタおよびメモリの中で物理的な（電子的な）量として表現され、コンピュータシステムのメモリあるいはレジスタ、またはその他のそのような種類の情報格納、転送または表示デバイスの中で物理的な量として同様に表現される他のデータへと変換される。

30 【0040】（セクション1）

（コンピュータシステム）本発明のアプリケーションプログラム（図4の210）、ハイレベルグラフィックスライブラリ220および230、およびローレベルハードウェア依存グラフィックスライブラリ（「HDL」）240は、本発明のコンピュータ制御されたグラフィックスディスプレイシステムに常駐する（reside）、コンピュータにより実行可能なインストラクションによって構成されている。これらの要素について、以下に詳しく述べる。図3は、本発明によるコンピュータ制御されたグラフィックスディスプレイシステムの一部として用いられるコンピュータシステム112の一例を示す。図3のコンピュータシステム112は、一例にすぎないので、本発明は、図3に示すものと同様に連携する同じ要素を備えていても備えていなくても、汎用コンピュータシステム、組み込みコンピュータシステム、およびグラフィックスディスプレイ専用のコンピュータシステムを含む多くの異なるコンピュータシステムにおいて動作させることができる。

50 【0041】図3のコンピュータシステム112は、情

18

報をやりとりするためのアドレス／データバス100と、情報およびインストラクションを処理するためにバス100と結合された中央処理ユニット101と、バス100と結合されて中央処理ユニット101のための情報およびインストラクションを格納するためのリード／ライトメモリ102（例えば、ランダムアクセスメモリやフラッシュメモリなどの他のリード／ライトメモリ）と、バス100に結合されてプロセッサ101のための静的な情報およびインストラクションを格納するためのリードオンリーメモリ103とを含んでいる。システム112は、また、バス100と結合されて情報およびインストラクションを格納するデータ記憶デバイス104（例えば磁気ディスクまたは光学ディスク、およびディスクドライブ）も含んでいる。システム112は、また、バス100と結合されて（あるいはバス100aを介して直接、ハードウェアユニット250と結合される）情報（例えばグラフィックスプリミティブ）をコンピュータユーザに表示するためのディスプレイデバイス105も含んでいる。システム112は、オプションとして、バス100と結合されて、情報およびコマンド選択を中央プロセッサ101に伝えるためのアルファニューメリック入力デバイス106（例えばアルファニューメリックおよびファンクションキーを含むデバイス）を含んでいてもよい。システム112は、オプションとして、バス100と結合されてユーザ入力情報およびコマンド選択をプロセッサ101に伝えるカーソルコントロールデバイス107を含んでいてもよい。システム112は、オプションとして、バス100と結合されてコマンド選択をプロセッサ101に伝える信号発生デバイス108を含んでいてもよい。プロセッサ101は、インストラクションまたはコードキャッシュ102aおよび（例えばRAMに特に設けられる）データキャッシュ102bを備えている。

【0042】本発明のコンピュータシステム112の中で利用される図3のディスプレイデバイス105は、ユーザに認識可能なグラフィックスイメージおよびアルファニューメリックキャラクタをつくるのに適した液晶デバイス、陰極線管、またはその他のディスプレイデバイスでありうる。オプションのカーソルコントロールデバイス107は、コンピュータユーザが、ディスプレイデバイス105のディスプレイスクリーン上の視認可能なシンボル（ポインタ）の2次元的な動きを動的に信号にすることができるようにする。カーソルコントロールデバイスの多くのインプリメンテーションとしては、トラックボール、マウス、タッチパッド、ジョイスティック、または与えられた向きの動き、あるいは変位のしかたを信号にすることができるアルファニューメリック入力デバイス106の特定のキーがこの技術分野ではよく知られている。カーソル手段107は、また、特定のキーおよびキーシーケンスコマンドを用いたキーボードか

19

らの入力を介して動かされたり、および／またはアクティベートされたりしてもよいことがわかるであろう。あるいはカーソルは、上に述べた複数の専用化されたカーソル指示デバイスからの入力を介して方向づけられたり、および／またはアクティベートされたりしてもよい。またバス 100 と結合されるか、あるいは、ディスプレイデバイス 105 と（バス 100 a を介して）直接に結合されるものとしては、高速グラフィックスレンダリング用のグラフィックスハードウェア（例えばグラフィックスアクセラレータ）ユニット 250 がある。グラフィックスハードウェアユニット 250 は、また、ビデオおよびその他のメモリ 102'（例えばディスプレイリストおよび／またはレジスタリングされた（registered）テクスチャマップを格納するための RAM）を含んでいてもよい。

【0043】図 3 は、本発明によるローレベルハードウェア依存グラフィックスライブラリ（HDGL）240 が RAM 102、ROM 103、記憶装置 104 の中に常駐しうることを示す。動作中には、HDGL 240 の一部は、コードキャッシュ 102 a の中に常駐することもできる。

【0044】図 4 は、本発明によるコンピュータ制御されたグラフィックスディスプレイシステム 200 のファンクションレイヤの論理表現である。ハードウェアユニット 250 およびディスプレイユニット 105 を除けば、図 4 の残りの要素は、コンピュータシステム 112（図 3）内に実行可能なインストラクションとしてインプリメントされ、RAM 102、ROM 103、記憶装置 104 の中に常駐でき、動作中は、HDGL 240 の一部は、コードキャッシュ 102 a の中にも常駐できる。ハイレベルアプリケーション 210（例えばシミュレータ、デザインツール、マルチメディアアプリケーション、医用画像アプリケーション、ゲームなど）は、ディスプレイスクリーン 105 上にイメージを生成することを要求するルーチンを含む。イメージは、グラフィックスプリミティブ（ポイント、ライン、ポリゴン、シェーディングされたポリゴン（shaded polygons）、ブレンドされたポリゴン（blended polygons）など）からなる。アプリケーション 210 のルーチンは、あるグラフィックスプリミティブの表示をリクエストし、グラフィックスプリミティブを表現するハードウェア非依存グラフィックス構造を供給することによって、ハイレベルグラフィックスライブラリ 220（3D-DDI）および／または 230（OPENGL）のグラフィックスレンダリングプロシージャにアクセスする。過去において、これらのグラフィックスライブラリ 220 および 230 は、ハードウェア非依存入力を用いていたが、その出力はハードウェア依存であり、それぞれのハードウェアユニット 250 について特化されたインプリメンテーションがサポートされることが要求された。グラフ

20

ィックスライブラリ 220 および 230 のハードウェア依存プロシージャを用いることは、この技術分野ではよく知られている。3D-DDI 220 および OPENGL 230 は、例示的であり、PC コンパチブル機および UNIX コンピュータを含む多くのコンピュータシステムで動作する。

【0045】本発明によれば、グラフィックスライブラリ 220 および 230 のグラフィックスレンダリングプロシージャ、ならびに、それらの入力および出力データ構造は、ハードウェア非依存である。これらのハイレベルライブラリ 220 および 230 は、グラフィックスハードウェアユニット 250 に特定（specific）であるローレベルハードウェア依存グラフィックスライブラリ

（HDGL）240 とインタフェースする。グラフィックスハードウェアユニット 250 は、グラフィックスアクセラレータ回路ボード、組み込み集積回路、または特別の目的のコンピュータシステム内の回路サブシステムなどである。ハードウェア非依存通信インタフェース 240 a は、ハイレベルグラフィックスライブラリ 220、230 と、HDGL 240 との間に要求される通信リンクを提供するために利用される。本発明によれば、多くのよく知られた通信インタフェースのどれでもインタフェース 240 a として用いることができる。

【0046】多種多様の異なるグラフィックスハードウェアユニット 250 を本発明の範囲内で用いることができる。本発明においては、HDGL 240 は、これらの異なるハードウェアユニット 250 に対応するように容易に適応可能であり、それでいながらハイレベルハードウェア非依存グラフィックスライブラリ 220 および 230 には、少しの変更しか必要ではないか、または全く変更を必要としない。

【0047】図 4 の HDGL 240 は、1 セットのハードウェア非依存グラフィックスレンダリングプロシージャを含むハイレベルグラフィックスライブラリ 220 および 230 とインタフェースする、1 セットのハードウェア依存ローレベルプロシージャを含む。インタフェースは、構造化された、しかしハードウェア非依存の入力データフォーマットを用いる。HDGL 240 の出力データフォーマットは、ハードウェア依存であり、ハードウェアユニット 250 に特定のものである。HDGL 240 のプロシージャは、非常に低いレベルで（例えばハードウェアユニット 250 に近くで）インプリメントされるので、限られた数の演算だけをサポートすればよい。HDGL 240 がローレベルであるので、よりハイレベルのグラフィックスライブラリ 220 および 230 を再設計する複雑な仕事（従来はなされていた）がなくなり、さまざまな異なるハードウェアユニット 250 に合わせてインプリメンテーションするために再設計することが容易であり、また特定のグラフィックスハードウェアユニットに合わせてインプリメンテーションするた

21

めに再設計することが容易である。グラフィックスライブラリ 220 および 230 は、ハードウェアに依存しないグラフィックスレンダリングプロシージャを含むので、本発明においては、これらのライブラリは、適用されうるそれぞれのハードウェアユニット 250 について再設計または再コーディングする必要がない。

【0048】本発明においては、ハイレベルグラフィックスライブラリ 220 および 230 は、本発明の HDGL 240 に依存して、イメージレンダリングのために必要なハードウェアに特定の機能を実現する。このように、図 4 のコンピュータ制御されたグラフィックスシステム 200 は、複雑さの緩和された（ローレベルの）HDGL 240 を変更することによってさまざまな異なるハードウェアユニット 250 に容易に適応させることができ、ハイレベルライブラリ 220 および 230 を変更することは必要でない。あるいは、多くの異なる HDGL 240 がシステム 200 内に含まれていてもよい。その場合、それぞれ異なる HDGL 240 を、特定のハードウェアユニット 250 に特定のものとすればよい。この場合の実施形態においては、本発明は、ユーザが特定のハードウェアユニット 250 の使用を選択することができ、選択されたハードウェアユニット 250 に対応する適切な HDGL が自動的にシステム 200 によって利用されることとなる。

【0049】アプリケーション 210 をコンパイルして、リンクするときには、ライブラリ 220 および 230 の要求されるプロシージャと、HDGL 240 のプロシージャおよび他の必要となる要素とがリンクされて、アプリケーション 210 の実行可能なかたちが形成される。

【0050】図 5 は、データフローの図であり、ディスプレイスクリーン上にイメージを生成するための、本発明によるコンピュータ制御されたグラフィックスディスプレイシステム 200 のレイヤ間での関連するデータフローを示す。データフローに伴う処理フローは、図 8 に示されている。処理フローについては、後で別途説明する。

【0051】図 5 を参照すれば、ハイレベルアプリケーション 210 は、表示のためのグラフィックスプリミティブまたはイメージを表すデータ構造を含むハードウェア非依存グラフィックスレンダリングリクエスト（「グラフィックスリクエスト」）を生成する。このハードウェア非依存データ構造 410 は、個々のグラフィックスプリミティブをレンダリングするためのデータ、またはビットレベル転送（BLT）や塗りつぶし（fill）などの他のグラフィックスレンダリングコマンドを含みうる。データ構造 410 は、単一のプリミティブまたは複数のプリミティブ、および／またはコマンドを備えうる。ハードウェア非依存データ構造 410 を、1 度に 1 つずつハイレベルグラフィックスライブラリ 220 およ

22

び 230 にわたすこともできるし、多数の個別のリクエストを、アレイフォーマットのかたちでライブラリ 220 または 230 に 1 度にわたすこともできる。

【0052】ハイレベルグラフィックスライブラリ 220 または 230 は、それぞれのグラフィックスリクエストに対応するハードウェア非依存データ構造 410 を受け取り、1 グループのデータ構造 410 がアプリケーション 210 から受け取られるまでそれらを集める。グループのサイズは可変であり、データ構造 410 に応答してハイレベルグラフィックスライブラリ 220 および 230 によって生成されるバッチアレイ 420 の利用可能なサイズに基づいて決定される。図 7 に示されているように、バッチアレイ 420 は、ハードウェア非依存であり、複数のバッチセル（例えば 420a、420b、420c など）から構成されるシーケンスを含んでいる。それぞれのバッチセルは、生成されるべき少なくとも 1 つのグラフィックスプリミティブ、および／または実行されるべきグラフィックスレンダリングコマンドを表現しており、オペランドおよび、そのオペランドに付随するデータのセットを含む。バッチアレイ 420 の内のバッチセルの個数は可変であり、ユーザによってプログラムされうる。いったんバッチセルの特定の個数、例えば x が決定されると、ハイレベルグラフィックスライブラリ 220 は、x 個のバッチセルが集まるか、他の時間的に重要な点に達するかするまで、特定のバッチアレイ 420 をメモリ 102（図 3）の中に構築する。

【0053】図 5 を参照すると、いったんバッチアレイ 420 がメモリ（例えば 102）中に構築されると、そのアレイは、本発明のハードウェア依存 HDGL 240 に転送される。HDGL 240 は、バッチアレイ 420 に含まれる複数のセルをシーケンシャルに処理する。それぞれのセルについて、ハードウェア非依存データ構造、およびそのバッチセルのオペランドが、ハードウェア依存ディスプレイリスト 430 に加えられるハードウェア依存マイクロインストラクションに翻訳される。ディスプレイリスト 430 は、メモリ（例えば 102）にも格納される。ディスプレイリスト 430 中のマイクロインストラクションは、ハードウェアユニット 250 によって、グラフィックスプリミティブおよび／またはグラフィックスレンダリングコマンドをディスプレイスクリーン 105 上にレンダリングするために用いられる。グラフィックスコマンドおよびアプリケーション 210 によって発生されたプリミティブをバッチアレイ方式で処理することによって、本発明による HDGL 240 のハードウェアに特定のレンダリングプロシージャは、本発明のシステム 200 内で利用可能なデータおよびコードキャッシュリソースを効率的に利用することができる。

【0054】図 6 は、本発明の HDGL 240 の主要要素を示す論理ブロック図である。HDGL 240 は、付

23

随するデータ構造（ブロック310）と組み合わせて用いられる1セットのローレベルプロシージャ（ブロック320および330）を含んでおり、メモリ中のディスプレイリスト（システム112内に常駐していてもよく、またはハードウェアユニット250内で専用であってもよい）内にマイクロインストラクションを生成することによって、直接、ハードウェアユニット250とインタフェースする。HDGL240は、グラフィックスイメージ、つまりレンダリングされるべきイメージに関するグラフィックス情報（ハードウェア非依存）をハイ

10 レベルグラフィックスライブラリ220および230から受け取り、ハードウェアユニット250にわたされ、ディスプレイ105上にイメージをレンダリングするために用いられるハードウェア依存ディスプレイリストをその情報から生成する。HDGL240によって受け取られたグラフィックスイメージをつくるためのグラフィックス情報は、典型的には、定義されたグラフィックスプリミティブおよびグラフィックスコマンドのかたちをとる。HDGL240によって生成されたディスプレイ

20 リストは、ハードウェア依存であり、グラフィックスイメージをつくるためにプリミティブをレンダリングするときにハードウェアユニット250によって用いられるマイクロインストラクションのリストである。

【0055】図6を参照すれば、HDGL240のデータ構造310は、グラフィックスプリミティブデータを、本発明において具体化される特定のハードウェア非依存フォーマットのかたちで受け取り、その他のオペランドおよびテクスチャマップを受け取るための入力構造を含む。ブロック320は、演算（operations）つまりハードウェア非依存グラフィックスコマンドおよびプリ

30 ミティブをハードウェア依存ディスプレイリストマイクロインストラクションに変換するための「パラメータ化」のセットを含む。以下に詳述するように、入力グラフィックスコマンドおよびプリミティブは、パッチセルに格納される。ブロック320は、システム200のレンダリングクオリティおよびレンダリングパフォーマンスを調整するためのプロシージャも含む。ブロック330は、レジスタリング（registering、例えば翻訳）、ローディングおよびテクスチャマップの表示のためのテクスチャマッピングプロシージャを含んでいる。ブロック330のレジスタリングプロシージャは、テクスチャマップデータを、カラーパレットへとインデクシング（indexing）するフォーマットと、RGB-アルファデータを用いるフォーマットとの間で翻訳するために用い

40 られる。HDGL240のインプリメンテーション例は、セクション2で説明される。

【0056】図8は、本発明のHDGL240によるレンダリング処理500の論理ブロックのフローチャートである。処理500は、例示的なコンピュータシステム112のようなコンピュータシステムの中でインプリメ

50

24

ントできることがわかるだろう。

【0057】論理ブロック510において、ハイレベルアプリケーション210は、1個のグラフィックスプリミティブの演算および／または1グループのプリミティブの演算、および／またはグラフィックスレンダリング演算が実行されるようリクエストをだす。これらのリクエストを表現するデータ構造410は、アプリケーション210から供給される。ハイレベルアプリケーション210は、1個のプリミティブを1度にリクエストすることもできる。あるいは、リクエストは、ある期間にわたってリクエストされたいくつかの個別のプリミティブおよび／またはグラフィックスレンダリング演算から構成されていてもよい。ブロック510のあいだにつくられるグラフィックスレンダリングリクエストのフォーマットは、ハードウェア非依存である。

【0058】論理ブロック515において、ハイレベルグラフィックスライブラリ220または230のハードウェア非依存グラフィックスレンダリングプロシージャは、プリミティブおよび／またはグラフィックスレンダリング演算を受け取り、ブロック510からの1個のリクエストに基づいて、またはシーケンシャルに受け取られたいくつかのリクエストに基づいて、パッチアレイ420を構築する。パッチアレイの許容サイズによっては、ブロック510から受け取られたリクエストを処理するために、個別のパッチアレイ420がいくつか必要になることがある。パッチアレイ420はメモリ102に格納される。そして、パッチアレイ420の最後のセルが、「パッチエンド」セルとしてマークが付けられる。構築されたパッチアレイ420の一部は、図9に示されているように、データキャッシュ102bに格納される。もしパッチアレイ420が十分に小さいのなら、パッチアレイ420の全体が、データキャッシュメモリ102b内に入ることになる。本発明においては、互いに異なるいくつかのメモリサイズのいずれを用いても効率よく動作可能ではあるが、データキャッシュメモリのサイズの一例を示せば、8キロバイト以上のオーダーとなる。パッチアレイ420のパッチフォーマットは、ハードウェア非依存である。

【0059】図8の論理ブロック520において、パッチアレイ420は、本発明のHDGL240へと転送される。本発明の範囲内では、ステップ520において実際にメモリ転送をおこなうことは必要ではない。その代わり、共用メモリ102内のパッチアレイ420のスターティングメモリ位置を示すポインタを、HDGL240へと転送すればよい。論理ブロック525においては、パッチアレイ420の各パッチセルを個別に処理することによって、コンピュータメモリ（例えば、メモリ102、またはハードウェアユニット250が直接にアクセス可能であるその他のメモリ）内のディスプレイリスト430に加えられるハードウェア特定のマイクロイ

25

ンストラクションをつくるために、HDGL 240のパラメータ化ルーチン（図5のブロック320）が用いられる。パラメータ化ルーチンは、「パッチエンド」に遭遇するまで、全パッチアレイ420においてループ状に割込みなしでおこなわれる。パラメータ化ルーチン320は、図9に示されているように、その全体がコードキャッシュメモリ102a内に入るように構成されている。

【0060】図9では、パッチアレイ420内のいくつかのパッチセルがデータキャッシュ102bにあり、かつ、HDGL 240のパラメータ化プロシージャ320がコードキャッシュ102a内にあるので、本発明によれば、これらのパッチセルに対して効率のよい処理メカニズムが得られる。なぜなら、図9に示されているデータについては、データキャッシュあるいはコードキャッシュのミスヒットに遭遇することがないからである。換言すれば、本発明においては、データの大半、およびパラメータ化をおこなうのに必要なコードのすべては、キャッシュメモリ内に配置されている。キャッシュ102b内のこれらのパッチセルが処理されている間に、ディスプレイリスト430が作成される。また、その時、HDGL 240のパラメータ化プロシージャ320によって、代表的なマイクロインストラクションがそれぞれのパッチセルについて、ディスプレイリスト430に加えられる。図8のブロック525に示されているように、メモリキャッシュ102bのすべてのパッチセルがHDGL 240によって処理されると、本発明では、図10に示されているように、別のグループをなすパッチセルをデータキャッシュ102b内にロードする。そして、このグループもまた、データあるいはコードキャッシュミスヒットを起こすことなく、パラメータ化プロシージャ320によって効率よく処理される。図11に示されているように、この処理は、パッチアレイ420のさらに別のグループをなすパッチセルについても反復される。

【0061】図8の処理525は、パッチアレイ420内でパッチエンドセルに遭遇するまで反復される。その時点で、ハードウェア依存ディスプレイリスト430が完成したとみなされる。論理ブロック530において、ディスプレイリスト430は、レンダリング用にハードウェアユニット250へと転送される。本発明の範囲内では、ステップ530において実際にメモリ転送をおこなうことは必要ではない。その代わり、共用メモリ102内のディスプレイリスト430のスターティングメモリ位置を示すポインタを、ハードウェアユニット250へと転送すればよい。ブロック530において、ディスプレイリストのマイクロインストラクションがハードウェアユニット250によって処理される。また、ビットマッピングされたイメージがディスプレイスクリーン105上に生成され、修正または上書きされるまでフレー

26

ムバッファあるいはその他のビデオメモリによって保持される。ハードウェアユニット250がディスプレイリスト430を処理している間に、プロセッサ101は、アプリケーション210のインストラクションを自由に処理することができる。

【0062】本発明のHDGL 240を通して、グラフィックスプリミティブおよび／またはグラフィックスレンダリング演算から構成されるパッチアレイ420を処理することによって、データおよびキャッシュメモリ102bおよび102aを、必要なインストラクションおよびデータの格納・供給に効率よく用いて、複数のパッチセルをシーケンシャルに処理することができる。この方法を用いれば、パラメータ化（例えば、ブロック525）をおこなう間に遭遇するデータキャッシュのミスヒットはごく少数となり、コードキャッシュのミスヒットに遭遇することはなくなる。

【0063】（レンダリングクオリティ／パフォーマンス調整）本発明のHDGL 240によれば、また、HDGL 240によっておこなわれるレンダリングパフォーマンス（例えば、速度）のレベルを変え、かつ、それに応じてイメージクオリティのレンダリングレベルをも同時に変えるための、ユーザが選択可能な調整をおこなうことが可能となる。具体的には、HDGL 240には、図12に示されているように、ユーザが調整可能なパフォーマンス／クオリティコントロールパネルを設けている。パフォーマンス／クオリティコントロールパネル610、または「ダイアル」は、本発明ではスクリーン105上に表示される。このパネル610は、ユニット106、カーソル107、あるいはその他の類似するスクリーンインタフェースを介するキーボードコントロールによって変えることができる、ユーザが調整可能な設定インジケータ615を有している。この設定インジケータ615は、部分610aによって示されているクオリティ設定を変えるために、ダイアル610に沿って調整可能である。そうすると、それに応じて、部分610bによって示されているように、パフォーマンス設定も変わる。ダイアル610は、クオリティおよびパフォーマンスの両方について最小設定値および最大設定値（ある実施形態においては、8ビット数の10進範囲を表す目盛り0～255が用いられる）を表示している。また、その他のダイアルフォーマット（例えば、円形ダイアルなど）を用いてもよい。

【0064】図12のクオリティおよびパフォーマンスの最小設定値および最大設定値は、これら2つの特性の間の逆の関係を説明するために、順番が逆になっている。ダイアル610によれば、レンダリングクオリティが上昇するにつれて、ユーザは、イメージクオリティつまりレンダリングクオリティが向上することを望む。この作用によって、レンダリングパフォーマンス値は自動的に低下する。なぜなら、所望のイメージクオリティを

27

実現するのに、グラフィックスシステム 200 がより多くの処理時間を要求するからである。逆に、レンダリングパフォーマンスが上昇すると、グラフィックスシステム 200 は、ハードウェアユニット 250 を通してグラフィックス情報をより高速に処理することを目的として、イメージクオリティのレベルを低下させる。実施形態の例を参照すると、セクション 2 では、設定インジケータ 615 の値を入力するのに用いられるプロシージャ Set Quality Dial が説明されている。

【0065】図 13 の論理処理 620 は、図 12 のクオリティ/パフォーマンスコントロールパネル 610 の設定インジケータ 615 に応答する HDGL 240 の処理を示している。処理 620 は、図 3 に示す種類のコンピュータシステムを用いてインプリメントされる。論理ブロック 625 は、設定インジケータ 615 の位置に基づいて、グラフィックスイメージのパーспекティブレンダリングのレベルを調整する。図 14 に示されているように、グラフィックス要素 650 は、線形モデルに基づいて、複数の再分割された領域 650a に再分割される。線形再分割は、計算の面では時間のかかるものではないので、システム 200 が、良好なレンダリングパフォーマンスを実現する高いパフォーマンスレートの処理をおこなうことを可能にする。しかし、図 15 に示されているように、グラフィックス要素 655 は、線形モデルに比較して、要素の 3 次元の向きをスクリーン 105 上により良好に描くことを可能にするパーспекティブモデルに基づいて再分割される。図 15 は、再分割された領域 655a が線形に分割されてはならず、要素 655 の 3 次元の向きに一部基づいていること、および、要素 655 の深さ方向（例えば Z 軸方向 657）に基づいて（パーспекティブを示すために）グラデーションがつけられていることを示している。パーспекティブ再分割をおこなうことによって、より高画質のグラフィックスイメージをレンダリングすることはできるが、膨大な量の計算を必要とするので、レンダリングパフォーマンスレートは低下してしまう。

【0066】図 13 の処理 625 は、設定インジケータ 615 が動かされてレンダリングパフォーマンスレートを増加させるにつれて、パーспекティブ再分割の量を減らし、線形再分割の量を増やす。同様に、処理 625 は、設定インジケータ 615 が動かされてレンダリングクオリティを増加させるにつれて、パーспекティブ再分割の量を増やし、線形再分割の量を減らす。本発明によれば、ある与えられた設定インジケータ 615 に基づくパーспекティブ/線形再分割の程度に到達するためには、いくつかある関数のうちのいずれを用いてもよい。ある実施形態においては、閾値を用いた判定がおこなわれる。その場合、もし設定インジケータ 615 がパフォーマンス (610b) の中間点を超えているのなら、再分割はすべて線形となり、もし設定インジケータ

28

615 がクオリティ (610a) の中間点を超えているのなら、再分割はすべてパーспекティブとなる。他の実施形態においては、パフォーマンスレートが増加すると、パーспекティブ再分割の程度を犠牲にして、線形再分割の程度または回数が増分を加えられて増加することになる。逆のことが起こっている時には、クオリティが向上することになる。

【0067】図 13 の論理ブロック 630 は、設定インジケータ 615 に基づいて、ポリゴン（例えばトライアングル）誤差補正係数を調整する。図 16 に示されているように、2 つのトライアングル 660 および 670 が領域 675 においてオーバーラップしている時、オーバーラップ領域 675 を正確にレンダリングするために、誤差補正係数または項を計算するための専用のプロシージャが用いられる。本発明により用いられるこの誤差補正プロシージャおよび誤差補正係数は 1994 年 9 月 1 日に Thomas Dye によって出願された『Incremental Orthogonal Error Correction for 3D Graphics (3D グラフィックス用の増分直交誤差補正)』と題する米国特許出願第 08/299,739 号（代理人の事件整理番号 984128 US）に記載されている。この特許出願は、本発明の譲受人に譲渡されている。ブロック 630 によれば、もし設定インジケータ 615 がより高いレートのパフォーマンスを示しているのなら、本発明の HDGL 240 は、計算され、上述の処理において用いられる誤差補正係数をなくすか、または、その量を減らす。このような状態では、トライアングル 660 および 670 は、画質は低下しているが、良好なレンダリングパフォーマンスレートでレンダリングされる。これに対して、もし設定インジケータ 615 がより高いイメージクオリティを示しているのなら、本発明の HDGL 240 は、計算され、上述の処理において用いられる誤差補正係数の量を増加させるか、最大にする。このような状態では、トライアングル 660 および 670 は、画質は向上しているが、より低いパフォーマンスレートでレンダリングされる。

【0068】図 13 の論理ブロック 635 は、パーспекティブレンダリングを切り捨てる (cut off) のに用いられるプリミティブの閾値サイズ設定を制御する。すなわち、ある範囲に属する小さなサイズのプリミティブのイメージクオリティは、そのようにサイズが小さいことが原因で、パーспекティブレンダリング技術から大きな恩恵を受けることはできない。閾値サイズは HDGL 240 に保持される。HDGL 240 は、その閾値サイズを下回るすべてのプリミティブについて、パーспекティブレンダリングを切り捨てる。パフォーマンス/クオリティ設定インジケータ 615（図 12）がより高いイメージクオリティに合わせて設定される時には、本発明のブロック 635 によって保持される閾値サイズは

29

低下し、プリミティブの大半は、パースペクティブレンダリング技術を用いて表示される。パフォーマンス／クオリティ設定インジケータ 615 (図 12) がより高いレンダリングパフォーマンスレートに合わせて設定される時には、本発明のブロック 635 によって保持される閾値サイズは増加し、徐々に多数のグラフィックスプリミティブあるいはイメージが、パースペクティブレンダリング技術を用いて表示されることがなくなり、パフォーマンスを増加させる。本発明によれば、ある与えられた設定インジケータ 615 に基づく閾値カットオフサイズに到達するためには、いくつかの関数のいずれを用いてもよい。

【0069】(テキストチャマップフォーマットの翻訳) 本発明の HDGL 240 は、また、テキスト情報(「テキスト」)をレジスタリングするために、テキストチャマップフォーマット間で翻訳をおこなうことも可能にする。ある実施形態例では、本発明によるコンピュータシステムによりインプリメントされた翻訳プロシージャ 700 (図 17) では、2つのテキストチャフォーマットが用いられる。そのうち 1つのフォーマットは、各画素に赤、緑青およびアルファ値が与えられる RGB-アルファフォーマットである。第 2フォーマットは、各画素に対して、カラーパレットへのインデックス値が与えられるインデックスフォーマットである。デフォルトフォーマットは、ハードウェアユニット 250 によって採用されたフォーマットに依存する。上記 2つのフォーマットのいずれか、または、その他の何らかのフォーマットをデフォルトフォーマットとすることができる。プロシージャ 700 は、RGB-アルファフォーマットがデフォルトである実施形態の一例を示している。

【0070】処理 700 においては、論理ブロック 705 で、HDGL 240 は、ある特定のテキストチャフォーマットでオリジナルテキストチャ情報を受け取る。このオリジナルテキストチャ情報あるいはデータは、メモリ 102 に入れられる。論理ブロック 710 において、本発明では、所定のフラグをチェックすることによって、テキストチャ情報がインデックスフォーマットであるか(例えば、ある特定のカラーパレットに対してインデックスが付けられている)どうかを判定する。もしそうなら、処理は論理ブロック 715 へと進み、テキストチャデータがインデックスフォーマットから RGB-アルファフォーマットへと翻訳される。ブロック 715 では、テキストチャデータの各画素について、その画素に対応するカラーパレットから特定の色属性を得るためにインデックス値が用いられる。いったん色属性が見つければ、その赤、緑、青およびアルファ値が決定され、RGB-アルファフォーマットに記録される。その結果得られるテキストチャ情報は、その後、オリジナルテキストチャ情報として、メモリ 102 内の新しい位置、または同一の位置に保存

30

される(例えば、オリジナルのテキストチャ情報を上書きする)。そして、処理はブロック 720 へと進む。

【0071】ブロック 710 において、もしインデックスフォーマットがインデックスモードではなかったのなら、フォーマットは RGB-アルファであると仮定される。処理がブロック 720 に進むまでに、テキストチャはレジスタに格納される。そして、処理は論理ブロック 720 へと進み、テキストチャ情報が転送され、HDGL 240 のパラメータ化プロシージャ 320 で用いるために割り当てられる。ハードウェアユニット 250 によって、このブロックには、メモリ 102 (システムメモリ) から、ハードウェアユニット 250 が直接アクセス可能である(かつ典型的には、そのユニット内にある)メモリ 102' へとテキストチャをロードすることが伴うことがある。レジスタリングされたテキストチャ情報は、その後、セクション 2 に記載されている実施形態に説明されているように、本発明により認識可能である、いくつかのポリゴンレンダリング演算によって表示される。

【0072】図 17 に示されている翻訳処理 700 は、また、テキストチャ情報以外の、表示すべき何らかのグラフィックス情報用にもおこなわれることは理解された。例えば、インデックスフォーマットあるいは RGB-アルファフォーマットのいずれかでプリミティブに関わる頂点を指定することができるし、本発明では、与えられたデフォルトに応じて、2つのフォーマットの間で翻訳することができる(具体的な説明については、下記セクション 2 を参照のこと)。

【0073】(パラメータ化) 図 18 は、バッチアレイ 420 のバッチセルを問い合わせ(interrogate)、そのセル上においてパラメータ化をおこなうことによって、ディスプレイリスト 430 用のマイクロインストラクションを作成するために用いられる、HDGL 240 のコンピュータによりインプリメントされる論理プロシージャ 800 を示すフローチャートである。処理 800 は、図 8 の処理 525 に対応する。プロシージャ 800 の特定のインプリメンテーションは、後述のセクション 2 では BuildDisplayList と呼ぶことにする。プロシージャ 800 は、論理ブロック 805 からスタートする。このブロック 805 では、本発明は、グラフィックスプロシージャが初期化されているかどうかをチェックする(例えば、InitGraph が実行されたかどうかをチェックする)。もしそうでないのなら、処理は論理ブロック 855 を介して終了する。もしグラフィックスが初期化されているのなら、処理は論理ブロック 810 へと進み、バッチアレイ 420 からあるバッチセルが得られ、そのバッチセルからパラメータが問い合わせされる。ある実施形態においては(後述のセクション 2 に説明されているように)、各バッチセルは、オペランドフィールドと、セル内のデータに伴う頂

31

点の個数を示すナンバフィールドと、フラグフィールドと、各頂点に対応する（例えば色属性、座標などの）データ構造（またはそれへのポインタ）とを含んでいる。オペランドフィールドは、レンダリングされるべきプリミティブ、またはおこなわれるべきグラフィックスコマンドあるいは演算を定義する。

【0074】セルのオペランドによって、820～845のプロシージャの1つが、スイッチ論理ブロック815によってコールされる。パッチセルの各パラメータを読み出し、かつ、ディスプレイリスト430内に入れられるマイクロインストラクションをそれから生成する処理は「パラメータ化」と呼ばれ、論理ブロック820～845によっておこなわれる。もしオペランドがBLTを示しているのなら、第1のスクリーン領域と第2のスクリーン領域との間で、論理ブロック820によってビットレベル転送がおこなわれる。第1のスクリーン領域は2つのコーナー座標（corner coordinates）（第1および第2の頂点）によって識別され、第2のスクリーン領域は、1つの座標（第3の頂点）によって識別される。第1の座標内の画素は、次に、BLT演算によって第2の座標へとコピーされる。フラグフィールドに基づいて、テクスチャ情報を転送のソースとして用いることもできる。また、第2のフラグ設定は、転送と、スクリーン更新との同期をとることができる。BLT演算に基づいて、本発明のHDGL240は、ハードウェア依存ディスプレイリスト430内に適切なマイクロインストラクションを生成する。

【0075】もしブロック815からのオペランドがFILLオペランドであるのなら、論理ブロック825が実行され、2つのコーナー（第1および第2の頂点）によって指定されたスクリーン領域が、指定されたソースの情報で塗りつぶされる。塗りつぶし用に選択された色は、第1の頂点データ構造において設定された色属性から生じる。もしZバッファリングがパッチフラグにおいて設定されるのなら、Zバッファが、ナンバフィールドにおいて決められた値で満たされる。FILL演算に基づいて、本発明のHDGL240は、ハードウェア依存ディスプレイリスト430を用いて適切なマイクロインストラクションを生成する。

【0076】もし図18のブロック815からのオペランドがPOINTオペランドであるのなら、論理ブロック830が実行され、ディスプレイリスト430内にマイクロインストラクションを生成することによって、頂点データ構造内で規定されるいくつかの点を表示する。その間に、処理される点の個数は、ナンバフィールドにおいて示される。POINTS演算に基づいて、本発明のHDGL240は、ハードウェア依存ディスプレイリスト430内に適切なマイクロインストラクションを生成する。

【0077】もしブロック815からのオペランドがP

32

OLYLINEオペランドであるのなら、論理ブロック835が実行され、ディスプレイリスト430内にマイクロインストラクションを生成することによって、パッチセルによって規定されるラインまたは一連のラインを表示する。そのラインまたは一連のラインを含む複数のポイントは、頂点データ構造によって規定され、ポイントの個数はナンバフィールドにおいて規定される。サポートされるラインレンダリングモードは3つある。すなわち、リストモード、ストリップモードおよびファンモードである。モードは、フラグフィールドにおいて設定される。リストモードにおいては、1セットの個別のラインがレンダリングされ、頂点の各ペアが各ラインの終点を表す。ストリップモードにおいては、以前のラインの終点に第2のラインが補足される。第1のラインは、1対の頂点によって規定され、その後各ラインは、単一の頂点によって規定される。ファンモードにおいては、第1の頂点は、すべてのラインに用いられるある点（共通点）を規定し、その後各頂点は、第1の頂点からそのポイントへのラインを規定する。これらのモードのそれぞれにおいて、ナンバフィールドは、POLYLINE演算の頂点の個数を規定する。POLYLINE演算に基づき、本発明のHDGL240は、処理されたパッチセルに対応するハードウェア依存ディスプレイリスト430内において適切なマイクロインストラクションを生成する。

【0078】もしブロック815からのオペランドがPOLYGONオペランドであるのなら、論理ブロック840が実行され、ディスプレイリスト430内にマイクロインストラクションを生成することによって、パッチセルによって規定されるポリゴンプリミティブまたは一連のポリゴンを表示する。パッチセルにおいて規定される頂点の個数は、ナンバフィールドにおいて示される。サポートされるポリゴンレンダリングモードは3つある。すなわち、リストモード、ストリップモードおよびファンモードである。モードは、フラグフィールドにおいて設定される。リストモードにおいては、3つの頂点毎に1個のポリゴンが規定される。ストリップモードにおいては、最初の3つの頂点が第1のポリゴンを規定し、その後の各頂点は、以前のポリゴンの頂点のうち2つを共有する新しいポリゴンを規定する。ファンモードにおいては、頂点データ構造における第1の頂点が各ポリゴンによって共有され、その後、2つの頂点毎に、新しいポリゴンが規定される。POLYGON演算に基づき、本発明のHDGL240は、処理されたパッチセルに対応するハードウェア依存ディスプレイリスト430内において適切なマイクロインストラクションを生成する。

【0079】もしブロック815からのオペランドが制御オペランドであるのなら、その制御オペランドを処理するために論理ブロック845が実行される。例えば、

33

制御オペランド **Saturate to Bounds** は、色制御レジスタを、カラーマスキングおよび飽和のためにナンバフィールドにおける第1および第2のバイトの値に対してハイおよびローに設定する。 **Set ZMask** オペランドは、衝突の検出とオブジェクトの識別のために、Zマスクを、ナンバフィールドの値に設定する。 **Set Display Page** 制御オペランドは、表示領域の開始オフセットを設定する。これは、ダブルおよびトリプルバッファリング用に用いられる。このオフセット値は、ナンバフィールドにおいてわたされる。セクション2に述べるように、その他のオペランドも使用可能である。

【0080】論理ブロック820~845が完了した後、ブロック850へと戻り、処理されたパッチのフラグフィールドが、**Batch_End** フラグを含んでいるかどうかを判定するためにテストされる。もしそうであるのなら、パッチアレイ420が完成していることになるので、実行は、ブロック855を介して終了する。もし **Batch_End** フラグが設定されていないのなら、処理はブロック810へと戻り、パッチアレイ420の次のパッチセルをフェッチして処理する。図18の処理によって生成されたディスプレイリストは、システム200がフラッシュディスプレイリストコマンドを受け取った時に、ディスプレイユニット105上にレンダリングされる（実施形態の例については、セクション2を参照のこと）。

【0081】処理800に必要なとされるこのインストラクションセットは十分にコンパクトであるので、大半のコンピュータシステムの大半のコードキャッシュメモリ102aにフィットする。このようにして、処理800を通して、パッチセルから構成されるパッチアレイ420を実行しても、コードキャッシュミスヒットが起こらないので、パラメータ化処理800を迅速に実行することができる。パッチアレイ420内で割込みなくパッチセルをシーケンシャルに実行することによって、パラメータ化をおこなう際にデータキャッシュ102bを最大限に使用することができる。

【0082】（セクション2）**HDGL240**の目的は、ハードウェア非依存グラフィックスライブラリ220および230に対して、容易に適応可能なハードウェア依存インタフェースを提供することである。ローレベルで異なるさまざまなバージョンの**HDGL240**を展開して、異なるさまざまなハードウェアユニットと共に動作させることができると予想される。あるいは、いくつかのハードウェアユニットに対して1セットの**HDGL240**のインプリメンテーションの一例を以下に述べる。本発明による**HDGL240**の範囲および着想内ではいくつかの代わりとなる実施形態を実現することがで

34

きるが、以下に示す実施形態の一例は、「C」言語を用いてモデリングされたものとして示すことにする。具体的には、図5の**HDGL240**のデータ構造ブロック310および演算ブロック320のインプリメンテーション例を示すことにする。当業者は、**HDGL240**に伴う具体的な構造およびプロシージャを用いて、所望の結果を得ることできるであろう。

【0083】（ハードウェア依存グラフィックスライブラリのデータ構造310の一例）以下に基本的なフィールドタイプを定義する。

【0084】

int	32ビット符号付き整数
DWORD	32ビット符号なし整数
WORD	16ビット符号なし整数
BYTE	8ビット符号なし文字
FIX	32ビット固定小数点数、そのうち16ビットは整数で残りの16ビットは分数
BOOL	真(1)/偽(0) 整数

HDGL240は、パラメータ化レイヤとして機能する。そこでは、グラフィックスプリミティブ（ポイント、ラインなど）のハイレベル記述子420が処理され、また、ハードウェアレベルコードが生成されて、ディスプレイリスト430内に格納される。**HDGL**の入力パッチアレイ420は、複数の**Batch**構造から構成されるアレイとして定義される。それぞれのパッチセルは、少なくとも1つの**Batch**構造を含んでいる。**Batch**構造の一例（例えば、パッチアレイ420の1パッチセル用の構造）を以下に提示する。

【0085】**Batch**

DWORD	dwOp
DWORD	dwFlags
DWORD	dwN
WORD	wTextID
Vert*	pVert

上記**Batch**構造は、1パッチセル用のプロトタイプである。値「**dwN**」は、**Batch**構造に付随するプリミティブにおける頂点のポイント数を規定している。

「**dwOp**」は、**Batch**構造によっておこなわれる基本的グラフィックス演算を決定するものであり、以下に示す演算のいずれか1つでありうる。

【0086】1. 2D演算

オペランド**BLT**は、最初の2つの頂点によって規定される矩形領域をコピーする。例えば、第1の頂点は左上隅を規定し、第2の頂点は右下隅を規定する（両方とも含む）。この領域のデスティネーションは、パッチアレイの**Batch**構造における第3の頂点によって、その左上の座標として規定される。もしフラグ**TIMED_BLT**が設定されるのなら、おこなわれている**BLT**演算は、セルの**dwN**フィールドに格納されているラインナンバをビデオリフレッシュすることによってトリガさ

35

れる。もしTEXTUREがフラグに存在しているのなら、テクスチャナンバdwNがBLT演算のソースとなる。オペランドFILLは、スクリーンの矩形領域の色あるいはその他の属性を設定する。この領域は、最初の2つの頂点によって規定される。例えば、第1の頂点は左上隅を規定し、第2の頂点は右下隅を規定する（両方とも含む）。もしZバッファリングがdwフラグ内で設定されているのなら、Zバッファは、dwNフィールドからの値（例えば、下位の16ビット）で満たされる。通常、これは、一回のインプリメンテーションにおいては0×FFFFに設定される。フレームバッファ（例えば、ハードウェアユニット250によって共有されている、またはそのユニットがアクセス可能であるメモリユニット）を満たしている色値は、第1頂点の色フィールドから取られる。FILLオペランドは、0で満たされていることを認識し、ハードウェアユニット250によってサポートされる高速クリア演算を設定する。

【0087】2. 3D演算

POINTオペランドは、頂点アレイによってその座標が指されている1セットのポイントをレンダリングする。ポイント数は、dwNフィールドにおいて規定される。POLYLINEオペランドは、(dwN-1)個以下の頂点からなるポリラインプリミティブをレンダリングする。POLYGONオペランドは、dwN個の頂点に基づいてポリゴンレンダリングする。ポリゴンの正確なタイプは、POLY*フラグを介して指定される。

【0088】3. グラフィックスレンダリング制御演算 SATURATE_TO_BOUNDSオペランドは、ある実施形態においては、色比較レジスタを、0~255の範囲である「dwN」の第1のバイト(1sb)および第2のバイトの値に対してハイおよびローに設定する。これは、カラーマスキングおよび飽和プロシージャにおいて用いられる。SET_Z_MASKオペランドは、衝突検出およびオブジェクト識別のために、ZマスクをdwNの値（例えば、下位の16ビット）に設定する。オブジェクト識別のアンダレイヤ原理は、Z座標のいくつかの上位ビットを別個に設定することによって、オブジェクト識別ナンバを保持することである。ある実施形態においては、このパーティションは、それらのビットを0に設定し、残りのビットを1に設定することによって、ハードウェア保護される。その後、衝突検出時には、衝突したZ値が、そのビットストリングとつきあわせて調べられ、衝突しているオブジェクトの識別が示される。SET_DISPLAY_PAGEオペランドは、ディスプレイのオフセットの開始を設定する。これは、ダブルおよびトリプルバッファリング用に用いられる。このオフセットは、dwNフィールドにおいてわたされる。

【0089】Batch構造に関して、「dwFlag

36

s」フィールドは、追加的ないくつかの特徴を示している。BATCH_ENDフラグは、バッチ処理を終了するために、バッチアレイ420の最後のセルにおいて設定される。もし1個しかバッチセルがないのなら、BATCH_ENDは、セルのdwFlagsにおいて設定される。TIMED_BLTフラグは、BLT演算内に設定され、指定されたスクリーンリフレッシュラインと同期の取られた時限の(timed)BLT演算が実行される。

【0090】「dwFlags」フィールドは、また、用いるべき特定のポリラインプリミティブと、ポリゴンプリミティブレンダリングモードとを示す。ポリラインプリミティブは、マルチラインでありうるものであり、(1)LINE_LIST、(2)LINE_STRIP、または(3)LINE_FANのいずれか1つを指定するリスト、ストリップまたはファンとして規定される。ポリゴンプリミティブは、マルチトライアングル状でありうるものであり、(1)POLY_LIST、(2)POLY_STRIP、または(3)POLY_FANのいずれか1つを指定するリスト、ストリップまたはファンとして規定される。これらのレンダリングモード間の違いを以下に説明する。LISTは、3つのポイントを用いてポリゴン（例えば、トライアングルポリゴン）を規定する順番づけられた1セットのポイントである。第1のトライアングルポリゴンは、ポイント0、1および2によって規定される。第2のトライアングルポリゴンは、ポイント3、4および5によって規定される。第3のトライアングルポリゴンは、ポイント6、7および8によって規定される。以下も同様である。STRIPは、ポイント0、1および2を第1のトライアングルポリゴンとして用いる順番づけられた1セットのポイントである。第2のトライアングルポリゴンは、ポイント1、2および3によって規定される。第3のトライアングルポリゴンは、ポイント2、3および4によって規定される。以下も同様である。FANは、ポイント0を多数のトライアングルポリゴンの中心として用いる順番づけられた1セットのポイントである。第1のトライアングルポリゴンは、ポイント0、1および2によって規定される。第2のトライアングルポリゴンは、ポイント0、2および3によって規定される。第3のトライアングルポリゴンは、ポイント0、3および4によって規定される。以下も同様である。

【0091】もし演算がZバッファされるべきものであるのなら、「dwFLAGS」内で以下のフラグの1つが加えられる。(1)ZBUFFERは、通常のZ演算をおこない、ZALWAYSは、画素およびZの両方を書き込み、あるいはZMASKは、通常のZ演算をおこなうが、Zバッファを更新しない。さらには、もしシェーディングが望まれているのなら、GOURAUDフラグがポリラインおよびポリゴン演算に加えられる。AL

37

PHAフラグは、この特徴をサポートするある種のハードウェアユニットにおいて、アルファブレンディング用にポリゴン演算に加えられる。

【0092】TEXTUREフラグは、ポリゴンをテクスチャマッピングする。このようにテクスチャ化されたポリゴンには、また、パースペクティブコレクションをターンオンするPERSPECTIVEも加えられている。後に詳しく説明するように、値「wTexID」は、レジスタリングされたテクスチャのテクスチャID数を表す。以下のフラグがテクスチャと共に用いられる。 (1) TEX_MAX_NEAREST (共通デフォルト)、あるいは (2) TEX_MAX_LINEAR、ならびに、TEX_MIN_NEARESTおよびTEX_MIN_LINEARのうちの1つ (別の共通デフォルト)。フラグTEX_TRANSPは、トランスペアレントなテクスチャに加えられる。トランスペアレントであるものとして扱われている色は、低い値および高い値の両方としてSTURATE_TO_BOUNDS演算により設定される。

【0093】HDGL240のもう1つのデータ構造は、グラフィックスプリミティブに対して頂点を規定するVert構造である。

【0094】

Vert

```

    FIX  x, y, z
    union c
        BYTE  index
        BYTE  r, g, b, a
    FIX  u, v, w

```

頂点の3D空間における位置は、x、yおよびz座標によって規定される。2つの異なるフォーマット、すなわち、「インデックス」フォーマット (ある実施形態においては8bppインデックス付きカラーモード用である) あるいは、要素として指定されるr、g、b、アルファの組み合わせのフォーマット (ある実施形態においては、16、24bppカラーモード用である) の間で、色を認識することができる。フィールドuおよびvは、テクスチャ空間における座標を規定する。また、wフィールドは、各種テクスチャ計算について同種の座標 (homogeneous coordinate) である。パッチセルにおけるそれぞれのグラフィックスプリミティブについて、これらのVert構造から構成されるアレイがメモリ102においてつくられ、Vertアレイのアドレスは、パッチアレイ420のパッチセルにおける「pVert」エントリを介してHDGL240にわたされる。この構造は、また、与えられたスクリーン座標に対して1対1でマッピングされるテクスチャマップ座標 (u、v) を容易に求めることができるようにする。これらの座標は、テクスチャマッピング演算において用いられる。パラメータ「w」は、頂点のパースペクティブ係数であ

38

る。その値は、ある実施形態では、頂点に対してパースペクティブが割り当てられていないことを意味する

「1」に設定され、また、テクスチャがその頂点に対してより大きく傾斜していることが原因で、より大きなパースペクティブ歪みがもたらされていることを意味する。「1」よりも大きな値に設定される。

【0095】HDGL240のもう1つのデータ構造としては、レジストレーション (registration) を目的とするテクスチャマップを規定するTexture構造がある。

【0096】Texture

```

WORD  wHeapID
WORD  wWidth
WORD  wHeight
BYTE*  pbTex
DWORD dwFlags

```

このデータ構造は、テクスチャの大きさフィールド (幅および高さ) と、システムメモリ102内に常駐しているテクスチャに対するポインタ (pbTex) とを有している。また、この構造は、関数TextureHeapAlloc () を用いているあるユーザのすべてのテクスチャについて割り当てられたヒープに対するハンドル (HeapID) も含んでいる。「dwFlags」フィールドは、テクスチャのタイプを保持している。ハードウェアユニット250次第で、このフィールドは、以下のいずれか1つとなる。すなわち、(1) 4bppインデックス付きテクスチャに対応するTEX_4BBP、(2) 8bppインデックス付きテクスチャに対応するTEX_8BBP、(3) トゥルーカラー565テクスチャに対応するTEX_16BBP、(4) トゥルーカラー888テクスチャに対応するTEX_24BBP、あるいはその他のカスタムセッティングである。

【0097】また、以下のフラグを「dwFlags」に論理的に加えることができる。すなわち、TEX_PROTECTであり、これは、テクスチャ認識時において、ユーザのテクスチャがそのハードウェアフォーマットによって上書きされないようにする。真のテクスチャについては、ある実施形態においては、テクスチャデータは、赤色を用いて3Dパケット化フォーマット [アルファ-BGR] で最下位バイトに書き込み、左から右へと線状に格納し、一番上のラインから一番下のラインへと格納することができる。インデックス付き8bppについては、1画素あたりわずか1バイトが必要になるだけであり、4bppについては、それぞれのバイト内に2つの画素をパックすることができる。ここで、そのペアの左側の画素は、上のほうのニブルに格納される。

【0098】HDGL240のまた別のデータ構造としては、1セットのパレットセルのタイプを規定するPalette構造がある。

【0099】Palette [256]

39

BYTE r
 BYTE g
 BYTE b

ある実施形態においては、この構造は、256個のパレットセルセットのタイプを、赤、緑および青の成分によって規定する。

【0100】HDGL240のまた別のデータ構造としては、スクリーン上の包括的矩形領域 (inclusive rectangular area) を規定するRect構造がある。

【0101】Rect

WORD x1
 WORD y1
 WORD x2
 WORD y2

ある実施形態においては、フォーマット (x1, y1) は左上隅を規定し、(x2, y2) は右下隅を規定する。HDGL240のまた別のデータ構造としては、Init構造がある。

【0102】Init

WORD resolution
 WORD color_mode
 WORD texspace

この構造は、初期化すべき解像度およびカラーモードを決定する。また、ロードされたテクスチャを格納するのに用いられるテクスチャ (プライベート) メモリの量も設定する。

【0103】HDGL240のさらに別のデータ構造としては、何らかの時点でのハードウェアユニット250の状態を示すDisplayContext構造がある。

【0104】DisplayContext

WORD wHardware
 WORD wVideoMemory
 WORD wTextureHeap
 WORD wTextureAvail
 DWORD fCapAlpha
 DWORD fCapTexture
 DWORD fCapZmask

このシステムワイド構造のアドレスは、別の初期化プロシージャをコールすることによって得られる。wHardwareフィールドは、下層のグラフィックスハードウェアのコードを含んでおり、サポートされたハードウェアバージョン (すなわち、(1) HARDWARE_A、HARDWARE_B、HARDWARE_Cなど) をリターンする。wVideoMemoryフィールドは、ハードウェアボード (Zバッファを含む) の上に存在しているビデオRAMの量を示す。wTextureHeapフィールドは、テクスチャに対して利用可能なメモリの総量を示し、wTextureAvailフィールドは、テクスチャの割り当てに用いることがで

40

きるメモリの量を示す。この値は、wTextureHeap以下である。なぜなら、異なる複数のユーザが、それぞれのテクスチャに対して既にヒープを割り当てている可能性があるからである。

【0105】(ハードウェア依存グラフィックスライブラリ演算320の一例) ブール値はいくつかの関数によってリターンされ、その関数が成功したか (TRUE)、あるいは失敗したか (FALSE) を信号で知らせる。失敗した場合には、GetErrCode () がコールされ、最後のエラーのエラーコードをリターンすることができる。また、GetErrMsg () がコールされ、ポインタを、最後のエラーを記述するゼロで終わるストリングへとリターンすることができる。

【0106】HDGL240のプロシージャにおいては、HDGL240を初期化するための他のどのプロシージャよりも早く、InitLib () がコールされる。別のプロシージャとしては、InitGraphプロシージャがある。

【0107】BOOL InitGraph (const WORD wResolution, const WORD wColorMode)

このプロシージャは、グラフィックスハードウェアユニット250を初期化し、システム200によってコールされて、初期化すべき解像度およびカラーモードを指定する。フィールド「wResolution」は以下のいずれか1つである。すなわち、(1) RES_GAMEは、解像度を640×480に設定する。(2) RES_STANDARDは、解像度を1024×768に設定する。「wColorMode」フィールドは、グラフィックスモード内での望ましいモードを、(1) 8bppパレット化に対応するCOL_8か、(2) 16bpp5-6-5に対応するCOL_16か、または(3) 24bppトゥルーカラー8-8-8に対応するCOL_24のいずれか1つに設定する。

【0108】ハードウェアユニット250次第では、「wResolution」および「wColorMode」の有効な組み合わせは、(1) 640×480×8、インデックス付き、ディスプレイポインタグル付きのdb、(2) 640×480×16、Yオフセット付きのtc、db、(3) 640×480×24、Yオフセット付きのtc、db、(4) 1024×768×8、インデックス付き、(5) 1024×768×46、tc、および(6) 1024×768×24、tcとなる。ここで「tc」は、True Colorモードの解像度を表し、「db」は、X86 CPUのインストラクションセットのためのデータバイトを表す。このプロシージャは、成功した場合にはTRUEをリターンし、そうでない場合にはFALSEをリターンする。可能なリターンエラーとしては、(1) ハードウェアユニット250のメモリテストが失敗した時のE_MEM

41

TEST、(2) ハードウェアユニット250の実行テストが失敗した時のE_ENGINE TEST、

(3) ハードウェアユニット250あるいはバス100のいずれかが存在していない時のE_PCI、(4) ハードウェアユニット250が内部レジスタテストに失敗した時のE_REGTEST、(5) モードが現在のハードウェア(例えば、ある特定のハードウェアユニット250に対する16bpp)によってサポートされていない時のE_NOTSUPPORTED、および(6) プロシージャに対してパラメータが無効である時のE_PARAMSがある。

【0109】HDGL240内のまた別のプロシージャとしては、Restore TextModeプロシージャがある。

【0110】Restore TextMode()
このプロシージャは、ビデオモードをVGAテキストモード(例えば、ある実施形態では、モード番号3)に復元する。このプロシージャは、InitGraphプロシージャに相当するプロシージャとして用いられるが、ハードウェアユニット250の状態は、このコールの後にも規定されていないままである。

【0111】HDGL240のまた別のプロシージャとしては、DisplayContext * QueryGraphicsDeviceプロシージャがある。

【0112】DisplayContext * QueryGraphicsDevice()
このプロシージャは、ハードウェアユニット250の状態に関する現在の情報を保持するシステムワイドのディスプレイコンテキスト構造のアドレスを得る。SetPaletteプロシージャは、HDGL240内のさらに別のプロシージャである。

【0113】BOOL SetPalette(WORD int_palette[, Palette * palette, WORD start, WORD count])

このプロシージャは、パレットレジスタを以下のセットのいずれか1つに初期化する。(int_paletteの値は)、(1) 階調パレット用のPAL_GREY、(2) インデックスカラー(3-3-2)をシミュレートするためのPAL_RGB、および(3) カスタムパレット用のPAL_CUSTOMがある。この場合、第3のパラメータがペーストされ、ユーザパレットに対するポインタとなる。引数「start」および「count」は、スターティングインデックス、およびある与えられたパレットから設定されるべきパレットセルの総数をそれぞれ規定する。パレット全体について、ある実施形態においては、これらの引数はそれぞれ、0および256となる。このプロシージャは、成功した場合にはTRUEをリターンし、そうでない場合にはFALSEをリターンする。また、可能なエラーとし

42

ては、無効パラメータに対するE_PARAMSがある。

【0114】SetQualityDialプロシージャは、HDGL240内のさらに別のプロシージャである。

【0115】SetQualityDial(BYTE bQuality)

このプロシージャは、クオリティ/パフォーマンスコントロールパネルの値を、ある実施形態では[0, 225]の範囲の値に設定することによって、レンダリングパフォーマンスおよびイメージクオリティを調整する。この値が低ければ低いほど、パラメータ化はさらに高速化されるが、精度は低下していく。もし現在のフレームがアニメーションステップであるのなら、より高い速度を実現するためには、この値をある小さな値(例えば、100未満)に設定するのが有効である。もし精密さが要求されているのなら、この値は、ある大きな値(例えば、200よりも大きな値)に設定される。

【0116】パラメータ化ルーチンは、典型的には、速度を最も重要な係数としてインプリメントされる。しかし、このようなアプローチではレンダリングクオリティが犠牲にされる。適正なバランスを実現するために、調整「クオリティ」が導入されている。これがゼロに設定されている時、レンダリングルーチンは、必ずしもグラフィックスの面では最も高精度であるとはいえない最高のパフォーマンスレートによるアプローチを用いることになる。この値を最大値にまで増加させることによって、精度は、パフォーマンスレートがいくらか損失されるという犠牲を払った上で向上する。精度は、ハードウェアユニット250次第で、グラフィックスプリミティブオーバーラップ領域と共に、Zおよび色計算におけるエラー項を反映する。テクスチャマッピングをおこなう時には、以下のヒューリスティックが用いられる。もしテクスチャをパースペクティブコレクションするのなら、そして、垂直方向のサイズが「小さい」あるいはZ方向の差が「小さい」のなら、パラメータ化ステップをより高速化するために、テクスチャは、パースペクティブ状ではなく、線状に描かれることになる。

【0117】バッチャアレイ420のハードウェア非依存バッチャセルを読み出し、メモリのディスプレイリスト430内に対応するハードウェア依存マイクロインストラクションを作成するプロシージャは、BuildDisplayListプロシージャである。

【0118】BOOL BuildDisplayList(Batch * p)

このプロシージャは、ポインタ(Batch * p)をバッチャタイプのバッチャアレイ420へと持っていく、システムメモリ102内、またはハードウェアユニット250のRAM内にディスプレイリストを作成する。これら一連のコマンドは、FlushDisplayList

43

がコールされて、ディスプレイリストをスクリーン105上にレンダリングするまでに、ディスプレイリストを作成する。このプロシージャは、成功した場合にはTRUEをリターンし、そうでない場合にはFALSEをリターンする。

【0119】(ハードウェア依存グラフィックスライブラリテクスチャマッピングプロシージャ330の一例)ある実施形態におけるHDGL240のテクスチャマッピングプロシージャを以下に説明する。

【0120】`BOOL TextureHeapAlloc (WORD* pwHeapID, WORD wHeapSize)`

プロシージャは、テクスチャヒープからメモリを割り当てる。このプロシージャは、マルチユーザホストが、互いに異なる処理を識別し、かつ、それらの処理固有のテクスチャヒープ空間を管理できるようにすることを意図するものである。「wHeapSize」フィールドは、ヒープサイズについてのユーザのリクエストを表す。このサイズは、`DisplayContext`構造において得られる値「wTextureAvail」以下でなければならない。このプロシージャは、成功した場合にはTRUEをリターンし、そうでない場合には、FALSEをリターンする。成功した場合、ポインタ「*pwHeapID」は、新しいヒープ空間に割り当てられるヒープハンドルを含んでいる。

【0121】HDGL240プロシージャである、`BOOL RegisterTexture (WORD* pwTexID, LL_Texture* pTex)`は、テクスチャをレジスタリングする。すべてのテクスチャ情報が、以前にユーザによって確立されたテクスチャデータ構造から読み出されるが、コールの後には必要なくなる。テクスチャデータは、ハードウェアユニット依存フォーマットに変換される。新しいデータが同一のメモリ内のソーステクスチャデータが位置するところに格納しなおされる(例えば、ソーステクスチャデータを上書きする)か、もし`TEX_PROTECT`ビットが「dwFlags」において設定されているのなら、あるいは新しいメモリブロックがそのビットに対して割り当てられているのなら、オリジナルのソーステクスチャデータが保存される。ユーザソーステクスチャに対するポインタ「pTex」は保存される。レジスタリングされるテクスチャのID番号は、pwTexIDの指すワードに格納される。このプロシージャは、成功した場合にはTRUEをリターンし、そうでない場合には、FALSEをリターンする。成功した場合、ポインタ*pwTexIDはテクスチャハンドルを含んでいる。また、もしソーステクスチャデータではなく、ハードウェア依存テクスチャフォーマットがプライベートライブラリヒープに格納されているのなら、`N_LIB_MEM`がdwFlagsにおいて設定される。

44

【0122】HDGL240プロシージャである、`BOOL FreeTexture (WORD wTexID)`は、そのレジストレーションによってリターンされたテクスチャナンバwTexIDを解放する。そのテクスチャはもはや用いることができないので、もし`IN_LIB_MEM`が、位置合わせプロシージャによってdwFlagsにおいて設定されたのなら、テクスチャライブラリヒープから解放される。

【0123】HDGL240プロシージャである、`BOOL LoadTexture (WORD wTexID)`は、そのハンドルがwTexIDであるテクスチャを、ハードウェアユニット250のテクスチャメモリ内へとロードする。テクスチャは、典型的には、wTexIDがBatch構造のwTexIDメンバにおいて用いられる前にロードされる。ある実施形態においては、テクスチャを数多くロードしたりアンロードした結果生じたメモリの断片化(fragmentation)が原因で、このプロシージャはエラーとなる可能性がある。この場合、LoadTextureを再試行して不要情報を整理するために、HeapCollectプロシージャが用いられる。しかし、ロードに失敗した場合には、不要情報の整理が自動的におこなわれるモードにライブラリを設定するために、MacroAutoHeapCollect (True)を用いることができる。このプロシージャは、成功した場合にはTRUEをリターンし、そうでない場合には、FALSEをリターンする。

【0124】HDGL240のプロシージャである、`BOOL UnloadTexture (WORD wTexID)`は、そのハンドルがwTexIDであるテクスチャを、ハードウェアユニット250のプライベートメモリからアンロードする。このプロシージャでは、テクスチャを「解放」せずに、テクスチャをレジスタリングされたままとし、後で再びロードできるようにする。このプロシージャは、成功した場合にはTRUEをリターンし、そうでない場合には、FALSEをリターンする。HDGL240プロシージャである、`HeapCollect (WORD wHeapID)`は、ハードウェアユニット250のプライベートメモリにおけるテクスチャデータの不要情報整理をおこなう。もしメモリの断片化が原因でLoadTextureが失敗したのなら、このプロシージャが用いられる。引数「wHeapID」は、断片化から復元されたヒープ(heap defragmented)のヒープハンドルである。このハンドルは、TextureHeapAlloc ()に対するコールによって得られる。

【0125】ヒープコレクトマクロもまた、HDGL240のこのインプリメンテーション、すなわち、AutoHeapCollect (WORD wHeapID, BOOL collect)によって用いられる。

【0126】このプロシージャは、自動ヒープコレクシ

45

オンをオンまたはオフに設定し、ハードウェアユニット 250 のプライベートテクスチャメモリ内へとテクスチャをロードする。引数「wHeapID」は、影響の及ぼされたヒープのヒープハンドルである。このハンドルは、TextureHeapAlloc () に対するコールによって得られる。

【0127】以上に本発明の好ましい実施形態、すなわち、柔軟で効率のよいグラフィックスレンダリングプロシージャをも含む、ハイレベルハードウェア非依存グラフィックスライブラリと、グラフィックスハードウェアユニットとの間のローレベルハードウェア依存グラフィックスライブラリによるインタフェース機能について説明した。特定の実施形態に基づいて本発明を説明したが、本発明は、そのような実施形態に限定されると解釈されるべきものではなく、むしろ添付のクレームによって解釈されるべきものであることは理解されたい。

【0128】

【発明の効果】本発明によれば、ハイレベルグラフィックスライブラリの再設計および再コーディングを必要とせず、異種のハードウェアユニットに対して容易に適用可能であるグラフィックスシステムを提供することができる。さらに本発明によれば、アプリケーションプログラムから発する複数のグラフィックスプリミティブからイメージをレンダリングするために、メモリリソースを効率的に利用できるグラフィックスシステムを提供することができる。

【図面の簡単な説明】

【図 1】ハードウェア依存型のグラフィックスライブラリ（例えば、3D-DDI および OPEN GL）をもつ従来技術によるグラフィックスディスプレイシステムを示す図である。

【図 2】グラフィックスプリミティブをレンダリングするための従来技術による処理のフローチャートである。

【図 3】本発明によるコンピュータ制御されたグラフィックスディスプレイシステム用のコンピュータシステムのブロック図である。

【図 4】ハードウェア非依存型のハイレベルグラフィックスライブラリ（例えば 3D-DDI および OPEN GL）をもつ本発明によるコンピュータ制御されたグラフィックスディスプレイシステムの各レイヤを示す図である。

【図 5】異なる複数のレイヤを含む本発明によるコンピュータ制御されたグラフィックスディスプレイシステムの各要素と、レイヤ間のグラフィックスデータ／情報のフローとを示すデータフローチャートである。

【図 6】本発明によるローレベルハードウェア依存グラフィックスライブラリ（HDGL、つまり「結合（binding）」ライブラリ）の各要素を示す図である。

【図 7】本発明によるバッチアレイの論理表現である。

【図 8】バッチセルのアレイを用いてディスプレイスク

46

リーン上にグラフィックスプリミティブを効率的にレンダリングするための、本発明による処理の処理フローチャートである。

【図 9】本発明によるバッチアレイ処理の第 1 フェーズのあいだのデータキャッシュメモリの内容、およびコードまたはインストラクションキャッシュメモリの内容を示す図である。

【図 10】本発明によるバッチアレイ処理の第 2 フェーズのあいだのデータキャッシュメモリの内容、およびコードまたはインストラクションキャッシュメモリの内容を示す図である。

【図 11】本発明によるバッチアレイ処理の第 3 フェーズのあいだのデータキャッシュメモリの内容、およびコードまたはインストラクションキャッシュメモリの内容を示す図である。

【図 12】本発明によるクオリティ／パフォーマンスコントロールパネルを示す図である。

【図 13】クオリティ／パフォーマンスコントロールパネルの設定に基づいてレンダリングクオリティ対レンダリングパフォーマンスを調整するための本発明による処理のフローチャートである。

【図 14】線形再分割を用いて再分割されたグラフィックスエレメントのグラフィックス表現を示す図である。

【図 15】パースペクティブ再分割を用いて再分割されたグラフィックスエレメントのグラフィックス表現を示す図である。

【図 16】2つのトライアングルポリゴンの間のオーバーラップ領域を示す図である。

【図 17】複数のテクスチャマップ用のカラーモード間で翻訳するための本発明による処理のフローチャートである。

【図 18】バッチアレイのバッチセルの中に格納されたグラフィックスデータ／情報に基づいてディスプレイリストを作成するための本発明によるハードウェア依存処理のフローチャートである。

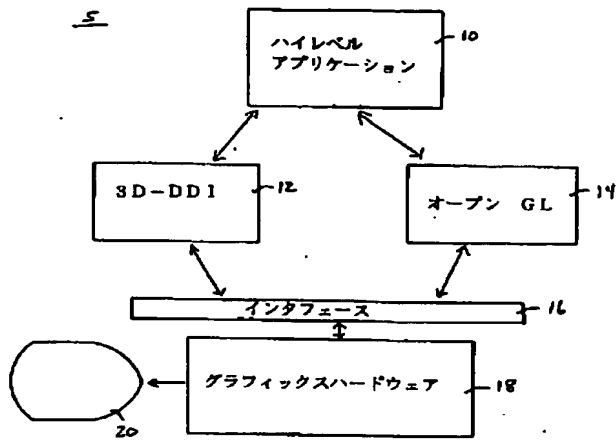
【符号の説明】

100 データバス
100a バス
101 プロセッサ
102 RAM
102a コードキャッシュ
102b データキャッシュ
103 ROM
104 データ記憶デバイス
105 ディスプレイデバイス
106 アルファニューメリック入力デバイス
107 カーソル手段
108 信号発生デバイス
112 コンピュータシステム
240 HDGL

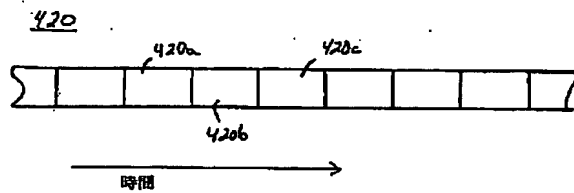
250 ハードウェアユニット

48

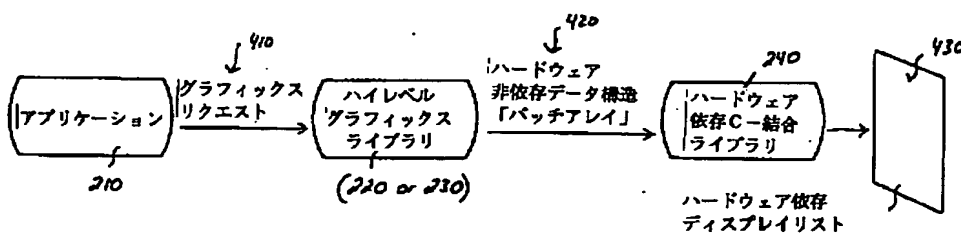
【図 1】



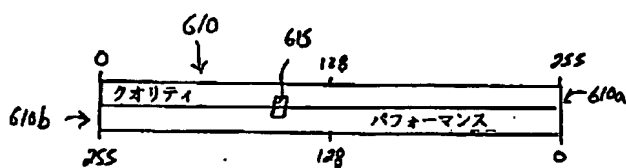
【図 7】



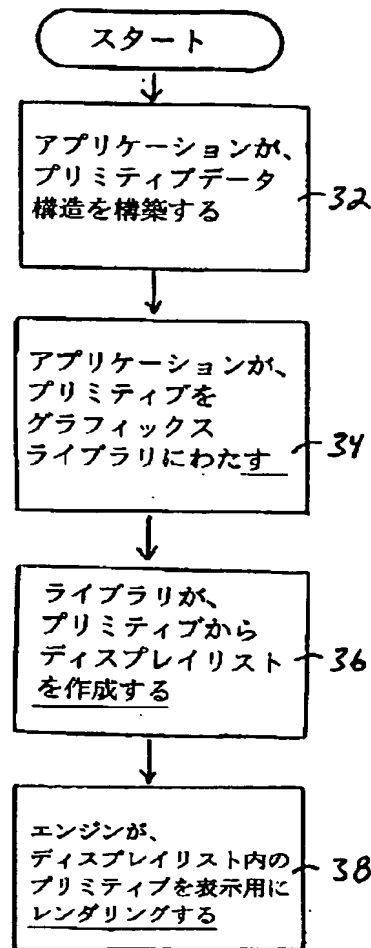
【図 5】



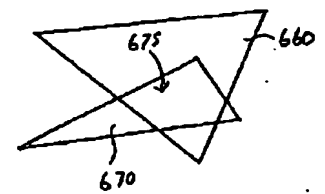
【図 12】



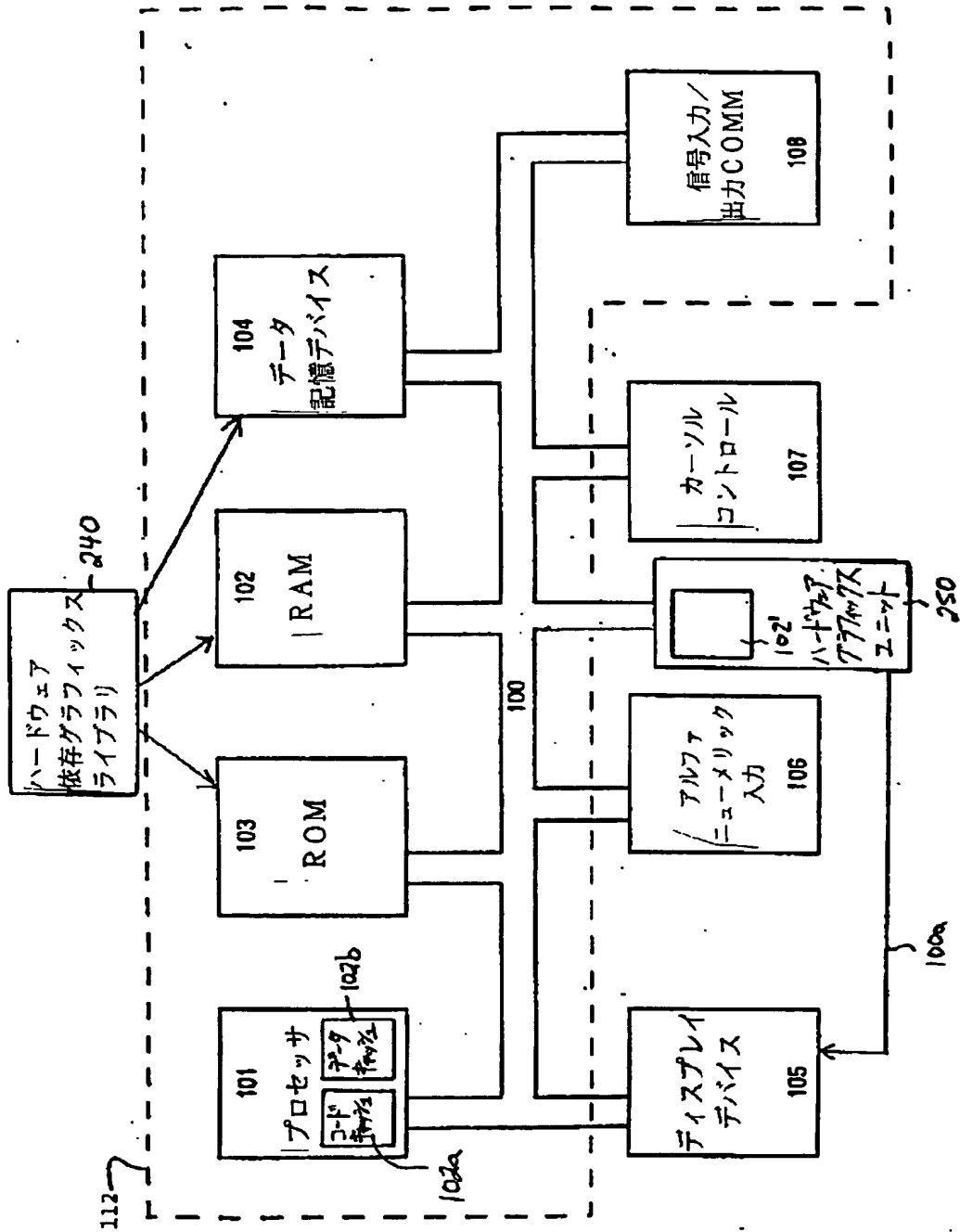
【図 2】



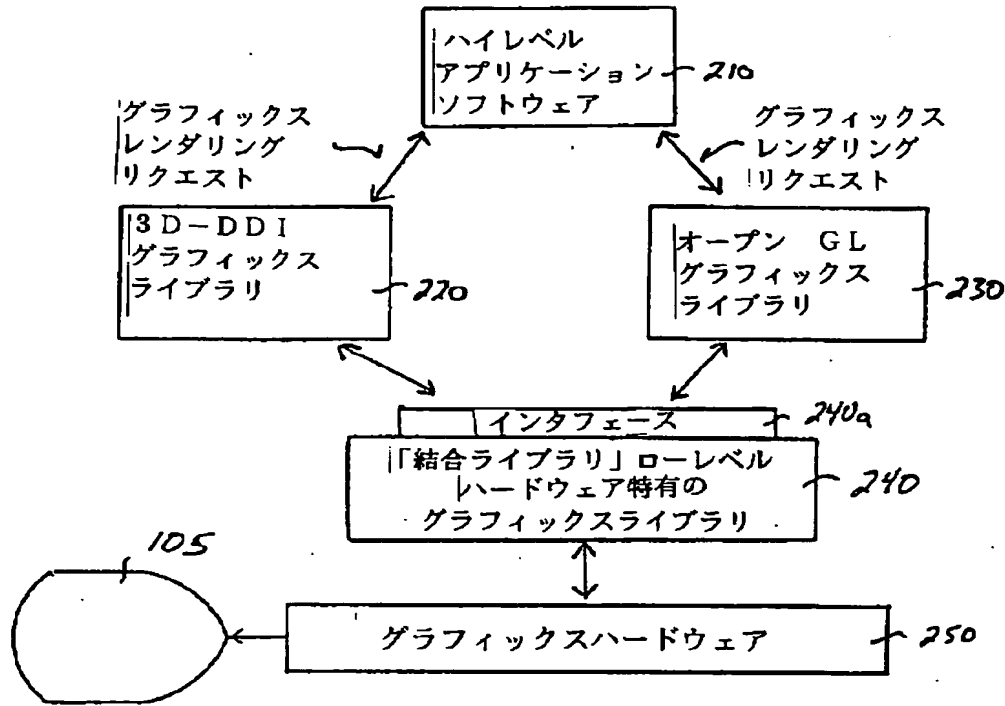
【図 16】



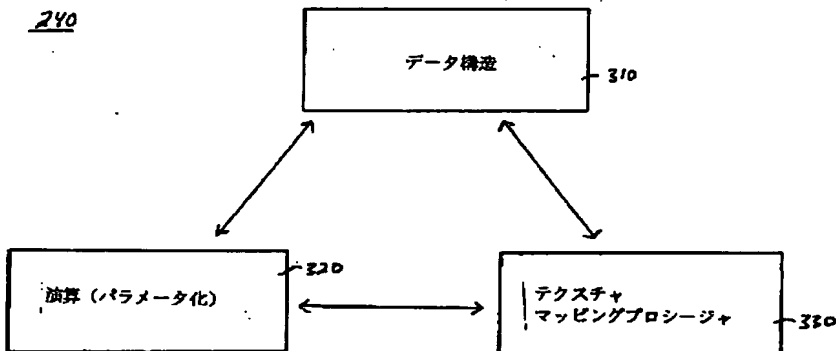
【図 3】



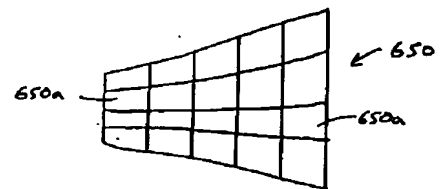
【図 4】



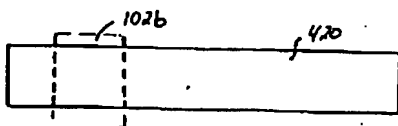
【図 6】



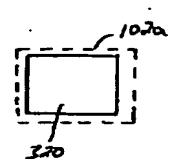
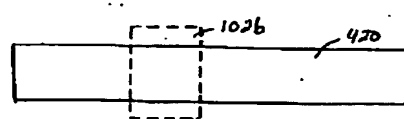
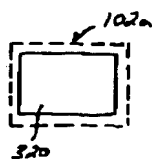
【図 14】



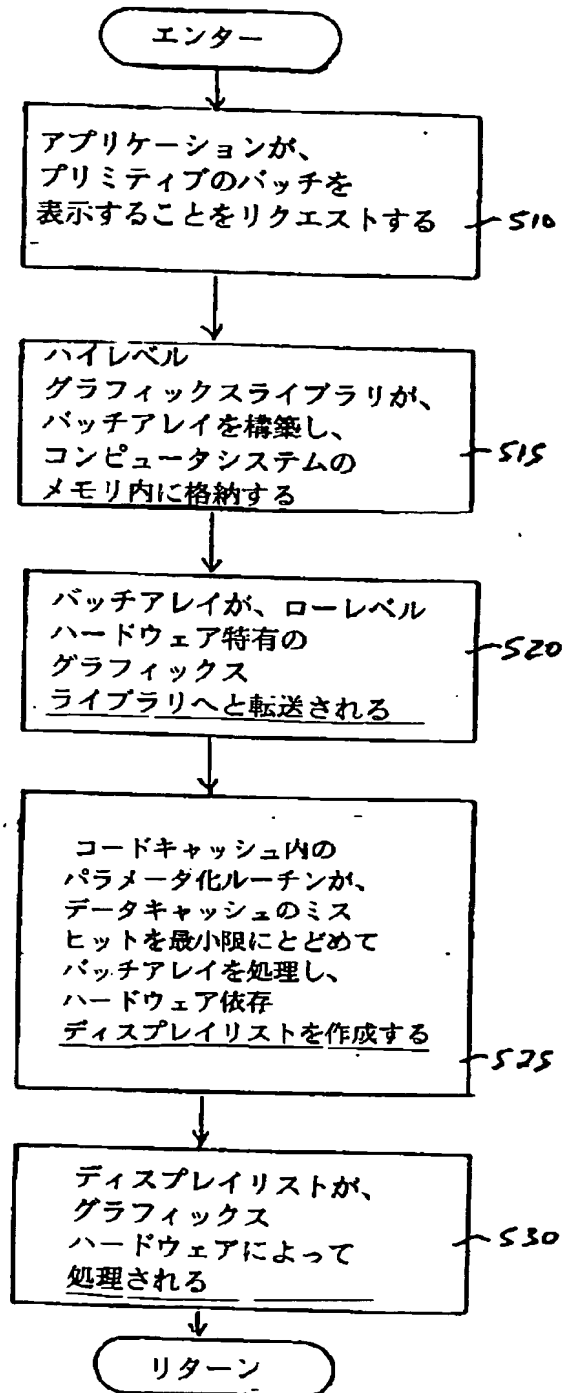
【図 9】



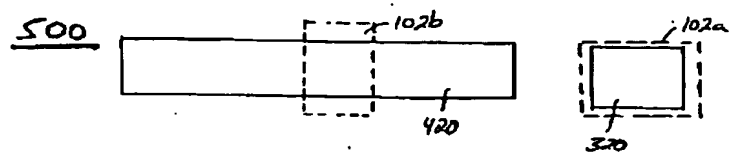
【図 10】



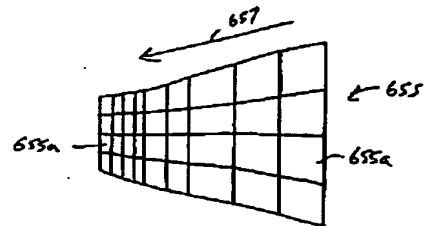
【図 8】



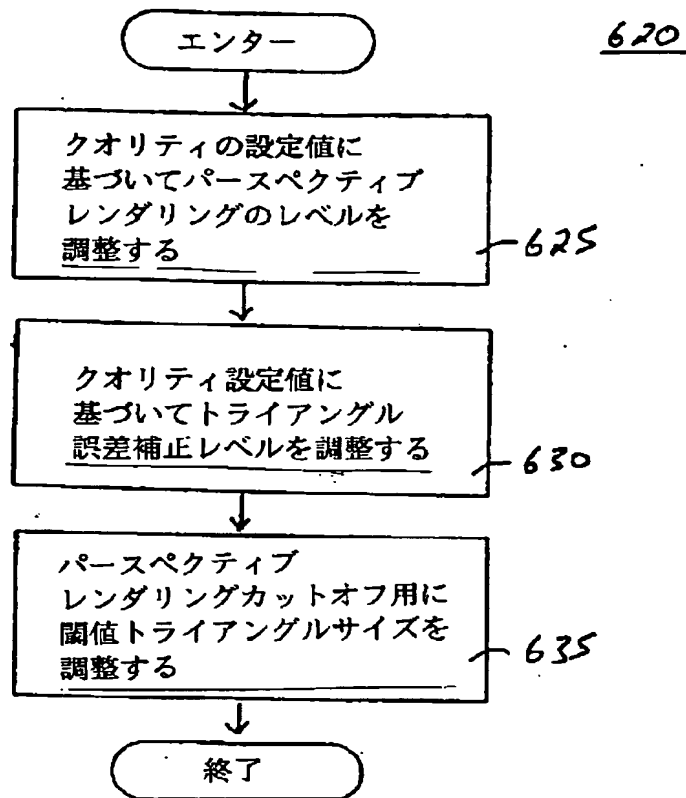
【図 11】



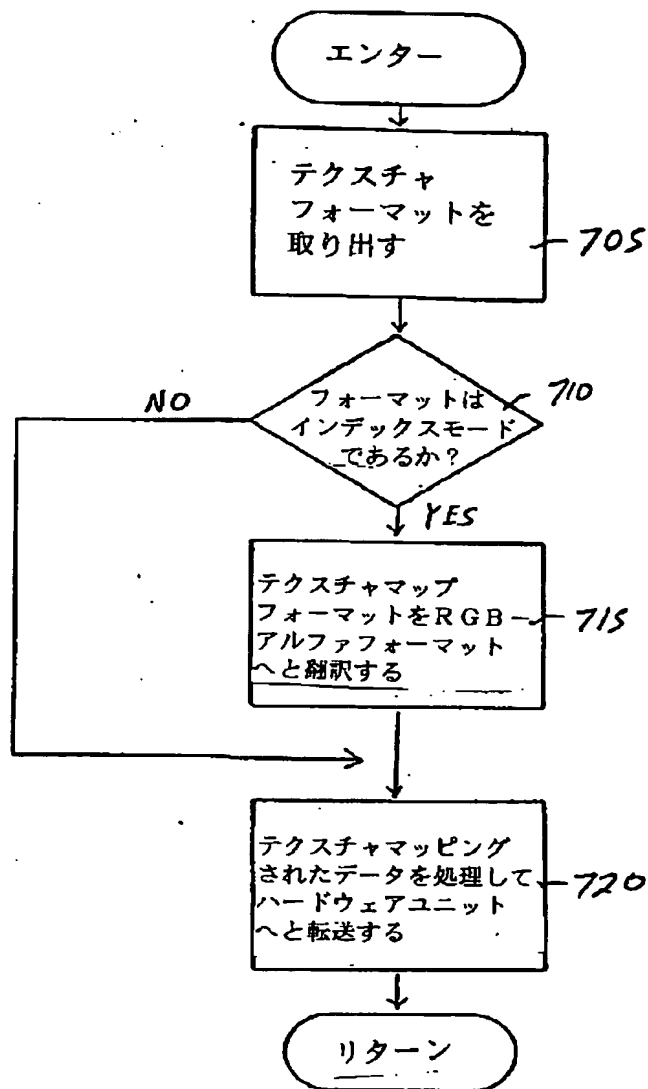
【図 15】



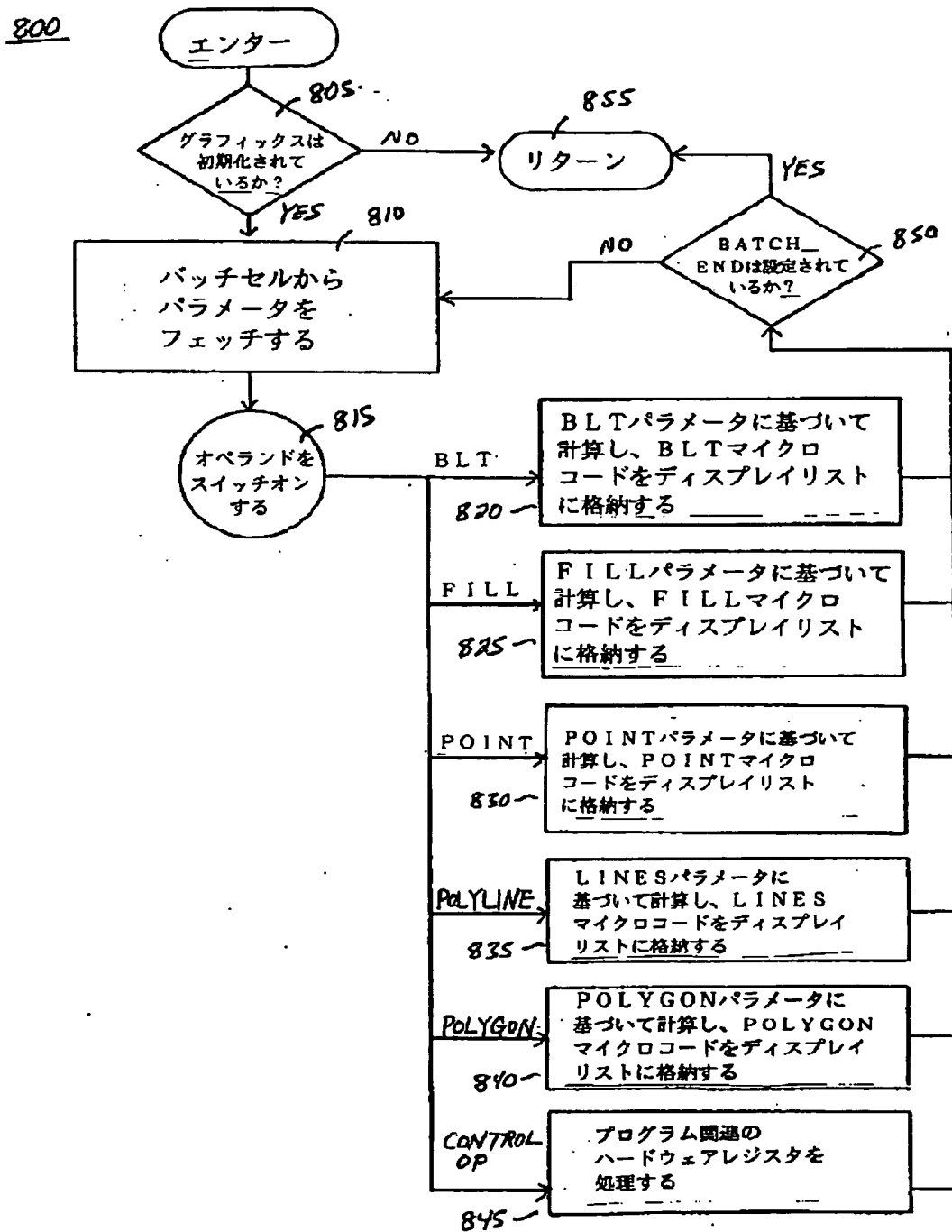
【図 13】



【図 17】



【図18】



フロントページの続き

(71)出願人 595158337

3100 West Warren Avenue,
Fremont, California
94538, U. S. A.